

## Proceedings

# 18<sup>th</sup> International Conference on Distributed Computing and Networking (ICDCN 2017)

## Conference Participants Copy



ICDCN 2017  
18<sup>th</sup> International Conference on Distributed Computing and Networking  
January 4 - 7, 2017 | Hyderabad, India



# Welcome Message from General Chair, ICDCN 2017

It is a pleasure to welcome you all to the 18<sup>th</sup> International Conference on Distributed Computing and Networking (ICDCN 2017) hosted by Institute for Development and Research in Banking Technology (IDRBT), Hyderabad at their campus. It is coincidental and apt that a premium networking and distributed computing conference is being held here when India is bidding to become a cashless transaction society with the help of innovative technologies.

The conference was inaugurated by Dr. Duvvuri Subbarao, former governor, Reserve Bank of India, who highlighted the role of technology in bringing innovations in banking transactions to keep them transparent without losing privacy.

ICDCN has been one of unique fora that provide a research synergy of distributed computing and networking for academia and industry. The conference attracts high quality submissions from around the world. This time there are 29 full-papers and 9 short-papers peer reviewed and accepted for presentation and inclusions in the proceedings, selected by the Technical Program Committee. The details of submission, process, countries, etc., are highlighted in the message from TPC Chairs. This year we have three distinguished keynote talks:

1. Prof. Chaitan Baru (NSF/USA, and UCSD), Harnessing the Data Revolution: A Perspective from the US National Science Foundation
2. Prof. Ashwin Gumaste (IIT Bombay), Is Network Hardware Virtualization the Holy Grail of SDN deployment?
3. Prof. Aniket Kate (Purdue University), Distributed IOWeYou Credit Networks

It is indeed a pleasure to thank the keynote speakers who have agreed to give keynotes on current frontier areas. Thanks go to them for their time, efforts, and elucidations.

On behalf of the Program Committee and Conference Organizers, I would like to express my gratitude to the Program Committee members, and the external reviewers who took part in the review process: their hard work and responsiveness has made it possible to stick to the tight review schedule and to arrive at a quality programme. We are confident that the authors of submitted papers will appreciate the reviewers' constructive comments and suggestions; we are sure that the reviews have enabled authors to further refine their works and in arriving at new openings/initiatives.

Apart from the conference, we had have three workshops and one doctoral symposium as briefed below:

- Workshop on Computing and Networking for Internet of Things (ComNet-IoT ) (5 selected out of 20 submissions)
- Workshop on Algorithms & Architectures for Distributed Data Analytics (AADDA) (2 selected out of 10 submissions)
- Workshop on Distributed Platforms and Technologies for Financial Services (DiPTeFS) (1 selected out of 4 submissions)

The entire work on these workshops was coordinated by Workshop Co-Chairs, Dr. Dilip Krishnaswamy (IBM Research, Bengaluru) and Prof. Sathya Peri (IIT Hyderabad). It is a pleasure to thank them for the excellent work done by them. The doctoral symposium was organized and coordinated by Dr. Padmalochan Bera (IIT Bhubaneswar) and Dr. Bivas Mitra (IIT Kharagpur). It is a pleasure to thank them for their wonderful efforts in getting excellent senior doctoral students and also for providing students a much needed feedback on their works, and current research challenges. IARCS ( Indian Association of Research in Computing Science) supported the participation of students in the symposium).

We are greatly indebted to IDRBT for the generous hosting and organization of this conference. The Steering Committee is indebted to Dr. A. S. Ramasastri, Director, IDRBT, for readily agreeing to host the conference with a very short notice and assigning the task of organization to Prof. G. R. Gangadharan, Convener, Center for Affordable Technologies at IDRBT. Prof. G. R. Gangadharan has done an excellent job of organization of the conference. On behalf of the Steering Committee and the TPC, I have a great pleasure in expressing my gratitude to him and the faculty and staff of IDRBT.

It is our pleasure to thank ACM SIGCOMM for according us ACM *In-cooperation status* and for agreeing to bring out the proceedings in ACM-ICPS. Thanks go to Ms. Anna Lacson from ACM who provided an excellent interface for bringing up the proceedings in ACM-ICPS. It is a pleasure to thank Dr. Vishwas Patil (IIT Bombay) for the support in bringing out the digital Proceedings.

From a practical perspective of conference-management, we would like to acknowledge that the whole process of paper submission, selection, and compilation in the proceedings was greatly simplified by the friendly interface of EasyChair conference system (<http://www.easychair.org>). We owe a lot to EasyChair's creators and maintainers for their commitment to scientific community and we are sure that they would continue to assist the users.

To conclude, we are confident that the 18th edition of ICDCN will contribute to enhance the knowledge in the broad areas of Distributed Computing and Networking and will further add to the prestige and reputation of ICDCN. We are sure that the readers will find the proceeding informative and valuable in their upcoming research and development activities.

Finally, we wish the participants a wonderful conference in Hyderabad and pleasant stay in India.

Prof. R. K. Shyamasundar, IIT Bombay  
General Chair

December 05, 2016.

# ICDCN 2017 Welcome Message from TPC Chairs

We are honored to welcome you to the main technical program of the 18<sup>th</sup> International Conference on Distributed Computing and Networking (ICDCN 2017). The main technical program of this year edition consists of two separate tracks – *Distributed Computing* and *Networking*. The Distributed Computing Track consists of 14 regular papers and 5 concise papers, chosen from a total number of 56 submitted papers. The Networking Track consists of 15 regular papers and 4 concise papers, chosen from a total of 53 submitted papers.

The conference followed a common submission process for all papers submitted to the main program. However, the review processes for the two tracks were handled separately, by the respective TPC chairs. For the *Distributed Computing* track, the review process was completed with the help of 29 expert TPC members, and two phases – a paper review phase followed by an online discussion phase. For the *Networking* track, the papers underwent a “quick reject” phase where a handful of papers which clearly did not meet the quality level of an acceptable paper for the conference were rejected (with one review) early in the reviewing process. The rest of the papers were reviewed by a total of 35 experts in the field, ensuring each paper receives 3-4 reviews. We are grateful to all TPC members for their tremendous effort in helping us put together a strong technical program!

We feel very proud bringing to you this high quality technical program, and we hope that you find the ICDCN 2017 technical program enjoyable and inspiring!

Paolo Bellavista (TPC Co-Chair – Networking Track)

Koushik Kar (TPC Co-Chair – Networking Track)

Sebastien Tixeuil (TPC Chair – Distributed Computing Track)

# Committee Members, ICDCN 2017

**CONFERENCE PATRON** A.S. Ramasastri, IDRBT Hyderabad, India

**GENERAL CHAIR** R.K. Shyamasundar, IIT Bombay, India

**PROGRAM CHAIR** *Distributed Computing Track*  
Sébastien Tixeuil, Université Pierre et Marie Curie -  
Paris 6, France

**PROGRAM CHAIRS** *Networking Track*  
Paolo Bellavista, University of Bologna, Italy  
Koushik Kar, Rensselaer Polytechnic Institute, USA

**ORGANIZING CHAIR** G.R. Gangadharan, IDRBT Hyderabad, India

**PUBLICITY CO-CHAIRS** Doina Bein, California State University, Fullerton, USA  
Stefano Chessa, University of Pisa, Italy  
Nirmala Roy, Univ. of Maryland Baltimore County, USA  
Yukiko Yamauchi, Kyushu University, Fukuoka, Japan

**WORKSHOP CO-CHAIRS** Dilip Krishnaswamy, IBM Research, India  
Sathya Peri, IIT Hyderabad, India

**PH.D. FORUM CO-CHAIRS** Padmalochan Bera, IIT Bhubaneswar, India  
Bivas Mitra, IIT Kharagpur, India

**STEERING COMMITTEE MEMBERS** Sajal K. Das, Missouri University of Science  
& Technology, USA (Co-Chair)  
Vijay Garg, University of Texas at Austin, USA  
Sukumar Ghosh, University of Iowa, USA (Co-chair)  
Dilip Krishnaswamy, IBM Research, Bangalore, India  
David Peleg, Weizmann Institute of Science, Isreal  
Michel Raynal, IRISA, France  
R.K. Shyamasundar, IIT Bombay, India

**PROGRAM COMMITTEE MEMBERS*****Distributed Computing Track***

Alysson Bessani, Univ. Lisboa, Portugal  
Anette Bieniusa, Univ. Kaiserslautern, Germany  
Silvia Bonomi, Univ. Roma "La Sapienza", Italy  
Sara Bouchenak, Insa Lyon, France  
Sylvie Delaët, Univ. Paris-Saclay, France  
Carole Delporte, IRIF, France  
Oksana Denysyuk, Univ. Calgary, Canada  
Shlomi Dolev, Ben Gurion Univ. of the Negev, Israel  
Paola Flocchini, Univ. Ottawa, Canada  
Eli Gafni, UCLA, USA  
Rachid Guerraoui, EPFL, Switzerland  
Michiko Inoue, NAIST, Japan  
Sayaka Kamei, Hiroshima Univ., Japan  
Partha Sarathi Mandal, IIT Guwahati, India  
Alessia Milani, Bordeaux Univ., France  
Achour Mostefaoui, IRISA, France  
Rui Oliveira, Univ. Minho, Portugal  
Merav Parter, MIT, USA  
Stacy Patterson, Rensselaer Polytechnic Institute, USA  
Sriram Pemmaraju, Univ. Iowa, USA  
Yvonne-Anne Pignolet, ABB, Switzerland  
Etienne Rivière, Univ. Neuchâtel, Switzerland  
Christian Scheideler, Univ. Paderborn, Germany  
Elad Schiller, Univ. Göteborg, Sweden  
Marteen van Steen, Univ. Twente, Netherlands  
Gadi Taubenfeld, The Interdisciplinary Center, Israel  
Amitabh Trehan, Queen's Univ. Belfast, Ireland  
Jennifer Welch, Univ. Texas A&M, USA  
Yukiko Yamauchi, Kyushu Univ., Japan

**PROGRAM COMMITTEE MEMBERS*****Networking Track***

Nirmala Shenoy, Rochester Institute of Technology, USA  
Biplab Sikdar, NUS, Singapore  
Pushpendra Singh, IIIT-Delhi, India  
Krishna Sivalingam, IIT Madras, India  
Subir Biswas, Michigan State University, USA  
Shikharesh Majumdar, University of Carlton, Canada  
Mun Choon Chan, NUS, Singapore  
Murat Yuksel, UNR, Reno, USA  
Chiu Chiang Tan, Temple University, USA  
Niloy Ganguli, IIT Kharagpur, India  
Vaskar Raychoudhury, IIT Roorkee, India  
Joy Kuri, Indian Institute of Science, India  
Baek-Young Choi, Univ. of Missouri - Kansas City, USA  
Zartash Afzal Uzmi, LUMS, Pakistan  
Nilanjan Banerjee, UMBC, USA  
Bhaskaran Raman, IIT Bombay, India  
Sachin Shetty, University of Tennessee, USA  
Pan Hui, Hong Kong Univ. of Science and Tech., Hong Kong  
Hong Xu, City Univ. of Hong Kong, Hong Kong  
Yanmin Zhu, Shanghai Jiao Tong Univ., China  
Mainak Chatterjee, University of Central Florida, USA  
Zubair Shafeiq, University of Iowa, USA  
Jianrong Cao, Hong Kong PolyU, Hong Kong  
Periklis Chatzisimios, ATEI, Thessaloniki, Greece  
Christos Douligeris, University of Piraeus, Greece  
Alfonso Ehijo, Universidad de Chile, Chile  
Alex Galis, University College London, UK  
Roch H. Glitho, Concordia University, Canada  
Burak Kantarci, Clarkson University, USA  
Axel Kuepper, TU, Berlin, Germany  
Toshikazu Nishimura, Ritsumeikan University, Japan  
Jukka Nurminen, Aalto University, Finland  
Antonio Puliafito, University of Messina, Italy  
Stefano Secci, LIP6 UPMC, France  
Sasu Tarkoma, University of Helsinki, Finland  
Anand Tripathi, University of Minnesota, USA  
Leye Wang, Institut Mines-Télécom/Télécom SudParis, France

## *Keynote Lecture 1*

# **Harnessing the Data Revolution: A Perspective from the US National Science Foundation**

Chaitan Baru

Computer and Information Science & Engineering Directorate  
National Science Foundation, USA  
[baru@sdsc.edu](mailto:baru@sdsc.edu)

**Abstract:** This talk will introduce NSF's vision for moving beyond initial, isolated approaches for data science research, services, and infrastructure, towards a cohesive, federated, national-scale approach to harness the data revolution and transform US science, engineering, and education over the next decade and beyond. The talk will highlight challenging technical issues in computer science as well as data science, and describe the role of distributed computing and networking in this context.

**Bio:** Chaitan Baru is Senior Advisor for Data Science in the Computer and Information Science and Engineering (CISE) Directorate at the National Science Foundation. He coordinates the cross-Foundation BIGDATA research program, advises the NSF Big Data Hubs and Spokes program, assists in strategic planning, and participates in interdisciplinary and inter-agency Data Science-related activities. He co-chairs the Big Data Inter-agency Working Group, and is co-author of the US Federal Big Data R&D Strategic Plan (<http://preview.tinyurl.com/z6w943j>), released in May 2016 under the auspices of the Networking and Information Technology R&D (NITRD) group of the National Coordination Office, White House Office of Science and Technology Policy. Baru is at NSF on assignment from the San Diego Supercomputer, University of California San Diego, where he is a Distinguished Scientist and Associate Director for Data Initiatives. He received his PhD and ME in Electrical Engineering from the University of Florida, and B.Tech in Electronics Engineering from IIT Madras.

## *Keynote Lecture 2*

# **Is Network Hardware Virtualization the Holy Grail of SDN deployment?**

**Ashwin Gumaste**

Department of Computer Science and Engineering

Indian Institute of Technology, Bombay  
Powai, Mumbai 400076  
[ashwin@ashwin.name](mailto:ashwin@ashwin.name)

**Abstract:** In this talk, we will consider the aspect of deploying SDN or software defined networks from a providers' perspective. SDN has shown promise of making the network programmable and hence agile towards offering new service paradigms that are specifically user-centric. SDN gear has the capability of performing complex network functions by bifurcating an SDN box into a control-plane that is centralized and programmable and a data-plane that can manifest any of the programmable functions with ease. Carrier-class performance requires guarantees in terms of latency, bandwidth support and other QoS measures. We examine what it would take for SDN "white-boxes" to meet carrier-class performance. Specifically, we will study our recently proposed network hardware virtualization concept and examine how it facilitates SDN white-boxes to be made service-centric. We will examine hardware architecture from the perspective of ASICs, FPGAs and network processors and understand the issues that determine data-plane abstraction. Metrics to evaluate such architectures and comparison with conventional products will be discussed. We will conclude by understanding a generic switch model that has been extended to an SDN white-box.

**Bio:** Ashwin Gumaste is currently an Associate Professor in the Department of CSE at IIT Bombay in Mumbai. He was the Institute Chair Associate Professor (2012-2015) and the JR Isaac Chair (2008-11) from 2008-10 he has been a Visiting Scientist at the Massachusetts Institute of Technology (MIT), Cambridge, USA. Ashwin has held positions with Fujitsu Laboratories (USA) Inc and has also worked with Cisco Systems and has been a consultant to Nokia Siemens Networks. He has also held short-term positions at Comcast, Lawrence Berkeley National Labs, and with Iowa State University. His work on light-trails has been widely referred, deployed and recognized by both industry and academia. His recent work on Omnipresent Ethernet has been adopted by tier-1 service providers and also resulted in the largest ever acquisition between any IIT and the industry. This has led to a family of transport products under the premise of Carrier Ethernet Switch Router. Ashwin has 23 granted US patents and has published about 150 papers in referred conferences and journals. He has also authored three books in broadband networks. For his contributions he was awarded the DST Swaranjayanti Fellowship in 2013, Government of India's DAE-SRC Outstanding Research Investigator Award in 2010, the Vikram Sarabhai research award in 2012, the IBM Faculty award in 2012, the NASI-Reliance Industries Platinum Jubilee award 2016, as well as the Indian National Academy of Engineering's (INAE) Young Engineer Award (2010).

*Keynote Lecture 3*

## Distributed IOWeYou Credit Networks

Aniket Kate  
Department of Computer Science  
Purdue University  
West Lafayette, IN 47907-2107, USA  
[aniket@purdue.edu](mailto:aniket@purdue.edu)

**Abstract:** IOWeYou (IOU) credit networks model transitive trust (or credit) between users in a decentralized environment. They have recently seen a rapid increase of popularity due to their flexible-yet-scalable design and robustness against intrusion. They serve today as a backbone of real-world permission-less payment settlement networks (e.g., Ripple and Stellar) as well as several other weak-identity systems such as spam-resistant communication protocols and Sybil-tolerant social networks. In payment scenarios, due to their unique capability to unite emerging crypto-currencies and user-defined currencies with the traditional fiat currency and banking systems, several existing and new payment enterprises are entering in this space. Nevertheless, this enthusiasm in the market significantly exceeds our understanding of security, privacy, and reliability of these inherently distributed systems. Currently employed ad hoc strategies to fix apparent flaws have made those systems vulnerable to bigger problems once they become lucrative targets for malicious players. In this talk, we will first define the concept of IOU credit networks, and describe some of the important credit network applications. We will then present and analyze the recent and ongoing projects to improve the credit-network security, privacy and reliability. We will end our discussion with interesting technical as well as legal challenges in the field.

**Bio:** Dr. Aniket Kate is an Assistant Professor in the computer science department at Purdue University. Before joining Purdue in 2015, he was a faculty member and an independent research group leader at Saarland University in Germany, where he was heading the Cryptographic Systems Research Group. He completed his postdoctoral fellowship at Max Planck Institute for Software Systems (MPI-SWS), Germany in 2012, and received his PhD from the University of Waterloo, Canada in 2010. His research interests include design, implement, and analyze privacy and transparency enhancing technologies. His research integrates cryptography, distributed computing, and trusted hardware. For more details, visit <https://www.cs.purdue.edu/homes/akate/>.

## Table of Contents

# ICDCN Conference Papers: Distributed Computing Track

### Session D1: Fault-tolerance

- Ashish Choudhury  
*Multi-valued Asynchronous Reliable Broadcast with a Strict Honest Majority*
- Luciana Arantes, Roy Friedman, Olivier Marin and Pierre Sens  
*Probabilistic Byzantine Tolerance Scheduling in Hybrid Cloud Environments*
- Ajoy K. Datta, Lawrence L. Larmore, Toshimitsu Masuzawa and Yuichi Sudo  
*A Self-Stabilizing Minimal k-Grouping Algorithm*

### Session D2: Locality and Mobility

- Klaus-Tycho Förster, Oliver Richter, Jochen Seidel and Roger Wattenhofer  
*Local Checkability in Dynamic Networks*
- (concise) Anisur Rahaman Molla and Gopal Pandurangan  
*Distributed Computation of Mixing Time and Local Mixing Time*
- Subhash Bhagat and Krishnendu Mukhopadhyaya  
*Fault-tolerant Gathering of Semi-synchronous Robots*
- (concise) Debasish Pattanayak, Kaushik Mondal, H. Ramesh and Partha Sarathi Mandal  
*Fault-Tolerant Gathering of Mobile Robots with Weak Multiplicity Detection*

### Session D3: Complexity

- Ali Mashreghi and Valerie King  
*Time-communication trade-offs for minimum spanning tree construction*
- Bapi Chatterjee  
*Lock-free Linearizable 1-Dimensional Range Queries*
- Sorrachai Yingchareonthawornchai, Vidhya Tekken Valapil, Sandeep Kulkarni, Eric Torng and Murat Demirbas  
*Efficient Algorithms for Predicate Detection using Hybrid Logical Clocks*

### Session D4: Security & Scheduling

- Xavier Vilaça and Luis Rodrigues  
*Accountability in Dynamic Networks*
- (concise) Varsha Bhat, Jaspal Singh and Sudarshan Iyengar  
*Secure Multiparty Construction of a Distributed Social Network*
- (concise) Umair Ullah Tariq and Hui Wu  
*Energy-Aware Scheduling of Periodic Conditional Task Graphs on MPSoCs*
- Sonal Kumari, Poonam Goyal, Ankit Sood, Dhruv Kumar, Sundar Balasubramaniam and Navneet Goyal  
*Exact, Fast and Scalable Parallel DBSCAN for Commodity Platforms*

### Session D5: Cloud Computing

- Shashikant Ilager and P.S.V.S Sai Prasad  
*Scalable MapReduce-based Fuzzy Min-Max Neural Network for Pattern Classification*
- Paolo Bellavista and Alessandro Zanni  
*Feasibility of Fog Computing Deployment based on Docker Containerization over RaspberryPi*

## Session D6: Resources

- Yiannis Georgiou, Emmanuel Jeannot, Adèle Villiermet and Guillaume Mercier  
*Topology-aware resource management for HPC applications*
- (concise) Hemant Tiwari, Prem Kumar and Balaji V Ramalingam  
*Optimized Offloading Using Local Clusters*
- Naveen Kumar and Anish Mathuria  
*Improved Write Access Control and Stronger Freshness Guarantee to Outsourced Data*

# ICDCN Conference Papers: Networking Track

## Session N1: Virtualization and Edge Computing

- Roy Friedman and David Sainz  
*An Architecture for SDN Based Sensor Networks*
- Xavier Vilaça, Luis Rodrigues, Joao Silva and Hugo Miranda  
*FastRank: Practical Lightweight Tolerance to Rational Behaviour in Edge Assisted Streaming*
- Fadi Alzhouri, Anjali Agarwal, Yan Liu and Ahmed Saleh Bataineh  
*Dynamic Pricing for Maximizing Cloud Revenue: A Column Generation Approach*

## Session N2: VANETs and Intelligent Transportation Systems

- Raj Jaiswal and Jaidhar Cd  
*PPRP: Predicted Position Based Routing Protocol Using Kalman Filter for Vehicular Ad-hoc Network*
- Chinmoy Ghorai and Indrajit Banerjee  
*A Novel Priority Based Exigent Data Diffusion Approach for Urban VANets*
- Sabin C C and Vaskar Raychoudhury  
*An Energy-efficient and Buffer-aware Routing Protocol for Opportunistic Smart Traffic Management*

## Session N3: Networking

- Gewu Bu and Maria Potop-Butucaru  
*Total Order Reliable Convergecast in WBAN*
- Sisi Duan, Sangkeun Lee, Supriya Chinthavali and Mallikarjun Shankar  
*Best Effort Broadcast under Cascading Failures in Interdependent Networks*
- Kopal Dwivedi, Akash Agarwal and Preetam Kumar  
*SER Performance of OFDM-Based AF and DF Cooperative Networks over Asymmetric Fading Channels*

## Session N4: Applications

- Divya Saxena, Vaskar Raychoudhury and Christian Becker  
*An NDNoT based Efficient Object Searching Scheme for Smart Home using RFIDs*
- Ioannis Katsidimas, Sotiris Nikoletseas, Theofanis Raptis and Christoforos Raptopoulos  
*Efficient Algorithms for Power Maximization in the Vector Model for Wireless Energy Transfer*
- Yayati Gupta, Sudarshan Iyengar and Neelika Kompala  
*Are We Birds of the Same Feather?*

## Session N5: Security and Coalitions

- Anshul Arora and Peddoju Sateesh Kumar  
*Minimizing Network Traffic Features for Android Mobile Malware Detection*
- Günter Fahrnberger  
*Contemporary IT Security for Military Online Collaboration Platforms*
- Jyotirmay Gupta, Prakash Chauhan, Madhabi Nath, Mekala Manvithasree, Sanjib K Deka and Nityananda Sarma  
*Coalitional Game Theory based Cooperative Spectrum Sensing in CRNs*

## Session N6: Concise Papers

- Divya Saxena, Vaskar Raychoudhury and Christian Becker  
*Implementation and Performance Evaluation of Name-based Forwarding Schemes in V-NDN*
- Sujata Swain and Rajdeep Niyogi  
*A Planning Based Approach For Context Aware Services Composition in Pervasive Systems*
- Suman Bhattacharjee, Siuli Roy, Sukumar Ghosh and Sipra Dasbit  
*Exploring the Impact of Connectivity on Dissemination of Post Disaster Situational Data over DTN*
- Saja Al-Mamoori, Subir Bandyopadhyay and Arunita Jaekel  
*Disaster-aware WDM network design for data centres*

# ICDCN Workshop Papers

## Session W1: Workshop on Computing and Networking for Internet of Things (ComNet-IoT )

- Jennath H S,Nandesh Nair, Sethuraman N Rao, Sujatha Narayanan, Dhanesh Raj, and Dilip Krishnaswamy  
*A Resilient Self-Organizing Offshore Communication Network for Fishermen*
- Dhanesh Raj, Vickram Parthasarathy, Sethuraman N Rao, Maneesha and Vinodini Ramesh  
*Performance Assessment of an Extremely Mobile Infrastructure Network over the Oceans*
- Arnab Kumar Ghoshal, and Nabanta Das  
*On Diameter Based Community Structure Identification in Networks*
- Joy Dutta, Chandreyee Chowdhury, Sarbani Roy, Asif Iqbal Midya, and Firoz Gazi  
*Towards Smart City: Sensing Air Quality in City based on Opportunistic Crowd-sensing*
- Rupam Some, Tuhina Samanta, and Indrajit Banerjee, Energy Aware  
*Cluster Head Load Balancing Scheme for Heterogeneous Wireless Ad Hoc Network*

## Session W2: Workshop on Algorithms & Architectures for Distributed Data Analytics (AADDA)

- Nandini Singhal, Sathya Peri, and Subrahmanyam Kalyanasundaram  
*Practical Multi-threaded Graph Coloring Algorithms for Shared Memory Architecture*
- Ashish Bhuker, Rajiv Misra, and Bhanu Pratap Singh  
*Balancing Maximal Independent Sets using Hadoop*

## Session W3: Workshop on Distributed Platforms and Technologies for Financial Services (DiPTeFS)

- Harika Narumanchi, and Nitesh Emmadi  
*Reinforcing Immutability of Permissioned Blockchains with Keyless Signatures' Infrastructure*

# Multi-valued Asynchronous Reliable Broadcast with a Strict Honest Majority

Ashish Choudhury<sup>\*</sup>  
 International Institute of Information Technology  
 26/C Electronic City, Hosur Road  
 Bangalore, India 560100  
 ashish.choudhury@iiitb.ac.in

## ABSTRACT

Asynchronous reliable broadcast is a fundamental primitive in secure distributed computing. In an asynchronous system of  $n$  mutually distrusting parties with  $t$  of them being Byzantine corrupted, reliable broadcast is possible if and only if  $t < n/3$ . With  $t < n/3$  there exists multi-valued asynchronous reliable broadcast protocol with optimal communication complexity of  $\mathcal{O}(n\ell)$  bits for broadcasting  $\ell$ -bit messages, provided  $\ell$  is sufficiently large. In this work, we give a multi-valued asynchronous reliable broadcast protocol with  $t < n/2$  and with communication complexity  $\mathcal{O}(n\ell)$  bits for sufficiently large  $\ell$ . To beat the lower bound of  $t < n/3$ , we deploy a non-equivocation mechanism, which can be instantiated using cryptography, along with a *trusted hardware set-up*. Such a mechanism prevents a corrupted party from sending conflicting messages to different (honest) parties. The applicability of non-equivocation mechanism has been investigated in the past to improve the fault-tolerance of secure distributed protocols in the asynchronous setting. This is the first work for deploying the non-equivocation mechanism to improve the communication complexity of asynchronous protocols.

## CCS Concepts

- **Security and privacy** → Cryptography; Distributed systems security; Security protocols;

## Keywords

Fault tolerance; secure distributed computing

## 1. INTRODUCTION

Reliable broadcast is a fundamental primitive in secure distributed computing [35]. Consider a distributed system of  $n$  mutually distrusting parties  $P_1, \dots, P_n$ , of which there is a special party called *sender*, denoted by  $S$ . A reliable broadcast protocol enables  $S$  to send some message  $m$  identically to all the  $n$  parties. Moreover,

---

\*Work supported by financial support from Infosys foundation.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ICDCN '17, January 04-07, 2017, Hyderabad, India

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007774>

this is ensured even if upto  $t$  parties (possibly including  $S$ ) behave maliciously and deviate from the ideal behaviour in any arbitrary fashion during the protocol execution. Reliable broadcast is used as an important primitive in several secure distributed-computing tasks, such as Verifiable Secret Sharing (VSS) [6, 10, 21, 23, 30, 37] and Secure Multi-party Computation (MPC) [3, 4, 6, 18, 24, 37].

Due to its fundamental importance, the broadcast problem has been studied rigorously in the past and several interesting results have been discovered regarding feasibility and communication complexity of broadcast protocols. In the synchronous setting, where parties are connected by pair-wise secure channels (which are private and authentic), broadcast is achievable if and only if  $t < n/3$  [7, 16, 35]; such broadcast protocols are also called *perfectly-secure* broadcast protocols and they are secure even against a computationally unbounded adversary. If we assume a set-up for information-theoretically secure signatures, then reliable broadcast is achievable for any  $t < n$  [36]. Such protocols are also secure against a computationally unbounded adversary, but may involve a negligible error in the protocol outcome. On the other hand if we assume a PKI set-up, then *cryptographically secure* broadcast is achievable with  $t < n$ , even if the parties are connected by pair-wise authentic channels (in this case the pair-wise private channels can be implemented using cryptography) [20]. Such protocols are secure against a computationally bounded adversary and may involve a negligible error in the protocol outcome.

**Multi-valued Broadcast:** From the seminal result of [19], it follows that any *perfectly-secure* broadcast protocol must involve a communication of  $\Omega(n^2)$  bits over the pair-wise channels to broadcast a single bit message. And it is a folk-lore belief that a broadcast protocol for broadcasting a *single bit* must involve a communication of  $\Omega(n^2)$  bits, irrespective of the setting. In several distributed computing applications such as MPC, there is often a requirement to broadcast long messages instead of a single bit. Such broadcast protocols are also known as *multi-valued broadcast*. One naive way of designing a multi-valued broadcast protocol is to invoke a dedicated instance of a broadcast protocol for broadcasting each individual bit of the message. If the message size is  $\ell$  bits, then broadcasting the whole message through this naive protocol will have communication complexity  $\ell \mathcal{BC}(1)$ ; here  $\mathcal{BC}(s)$  denotes the communication complexity for broadcasting an  $s$ -bit message. If broadcasting a single bit requires a communication of  $\Omega(n^2)$  bits, then clearly this naive protocol will have communication complexity  $\Omega(n^2\ell)$  bits, which is quadratic in the number of parties  $n$ .

An interesting and clever approach to improve the communication complexity of the naive multi-valued protocol is to design a broadcast protocol, where “most” of the communication is done over the point-to-point channels and the single-bit broadcast pro-

tocol is invoked to broadcast only “short” messages, whose size is *independent* of  $\ell$ . This approach was first proposed in [40] for the perfectly-secure setting, followed by several interesting multi-valued broadcast protocols [22, 27, 32, 34] in various settings. These protocols have communication complexity of the form  $\mathcal{O}(n\ell) + \mathcal{BC}(\text{poly}(n, \kappa))$  bits, where  $\kappa$  is some security parameter. Hence for “sufficiently large” values of  $\ell$ , these protocols have communication complexity  $\mathcal{O}(n\ell)$  bits, which is linear in  $n$  and which is asymptotically optimal in  $\ell$  [22].

We stress that the above multi-valued protocols are only *asymptotically* optimal in  $\ell$ . That is, they are better than the naive multi-valued protocol only when long messages are needed to be broadcast. Indeed there are applications like MPC where such long messages appear, for example when several gates are evaluated interactively by the parties in parallel (see for example [11–13], where multi-valued broadcast protocols have been deployed for broadcasting long messages to improve the communication complexity of MPC protocols).

**Asynchronous Reliable Broadcast:** All the above results are in the synchronous communication setting, where it is assumed that the parties are synchronised through a global clock and where strict upper bounds are assumed on the message delays. Such a communication setting does not model the real-world networks, like the Internet where messages can be arbitrarily delayed. Such networks are more appropriately modelled by the asynchronous communication setting [9]. The inherent difficulty in designing asynchronous protocols is that in a completely asynchronous setting, we cannot distinguish a corrupted sender from a slow, but honest sender. More specifically, if a party does not receive a message which it is supposed to receive as per the protocol, then the party cannot decide whether the sender is corrupted (and purposely not sending the message) or whether the sender is honest, but slow (whose message is arbitrarily delayed). As a result, no party can afford to receive communication from all the  $n$  parties, as this may turn out to be an endless wait. So as soon a party receives communication from  $n - t$  parties, it has to proceed to the next step of the protocol, ignoring communication from the remaining  $t$  parties. But in this process, communication from  $t$  potentially honest parties may be ignored. For an excellent introduction to asynchronous protocols, see [9].

In the asynchronous setting, information-theoretically secure broadcast as well as computationally-secure broadcast is possible if and only if  $t < n/3$  [8, 25, 26, 39]. So unlike the synchronous communication setting, the optimal resilience bound for asynchronous broadcast protocols *does not* vary across various settings. A very elegant construction of asynchronous broadcast with  $t < n/3$  is due to [8], which is perfectly-secure. The protocol has communication complexity  $\mathcal{O}(n^2\ell)$  bits to broadcast an  $\ell$ -bit message. In [33, 34], multi-valued asynchronous broadcast protocols with  $t < n/3$  are presented. These protocols have communication complexity  $\mathcal{O}(n\ell)$  bits (for sufficiently large  $\ell$ ), which is asymptotically optimal in  $\ell$ .

**Asynchronous Broadcast with  $t \geq n/3$ :** In [14], it is observed that if we somehow restrict the corrupted parties from sending “conflicting messages” to the honest parties (we call this as *equivocation*), then the resilience of several distributed computing tasks such as Byzantine agreement and reliable broadcast can be reduced in the asynchronous setting. We stress that the so called “non-equivocation mechanism” (namely the mechanism which prevents a corrupted party from equivocating) does not ensure that the message of a corrupted party reaches *all* the  $n$  parties; the mecha-

nism just ensures that the parties who receive the message from a corrupted party have obtained the identical message<sup>1</sup>. The work of [14] is followed by several interesting works [2, 15, 17, 28, 29, 31] which show varieties of practical ways of implementing the non-equivocation mechanism and how to design protocols with better resilience for various asynchronous distributed computing tasks such as asynchronous Byzantine agreement, asynchronous broadcast, asynchronous verifiable secret-sharing and asynchronous MPC.

One of the simplest ways to implement the non-equivocation mechanism is by introducing a small trusted hardware assumption, namely the presence of a small trusted hardware module with each party. The hardware module is just a trusted, increment-only local counter and a signature oracle. Conceptually, this provides once-in-a-lifetime attestation, and implements non-equivocation using the fact that the counter cannot be decreased. As a result, for every possible counter value  $c$ , there can be at most one message signed by the module. Using this arrangement, non-equivocation can be implemented as follows: we first assign unique counter values to each message that a party is supposed to communicate in a protocol (this can be done in an unambiguous fashion). Then every time a party is supposed to send some message in the protocol, it first gets the message and the corresponding counter value signed by the module, before sending it to the desired recipient(s). The receiver(s) on receiving the message verifies the signature (with respect to the pre-assigned counter value) and accepts the message if the signature is valid. The use of signatures (or transferable authentication) above complements non-equivocation by making it transferable as required in the asynchronous environment with unknown delays. The additional communication cost involved in this entire mechanism is that of sending the signatures, along with messages. In [15, 17] it was shown how to beat the  $t < n/3$  bound and design an asynchronous reliable broadcast protocol with  $t < n$ , assuming a setup for implementing transferable non-equivocation mechanism. The protocol has communication complexity  $\mathcal{O}(n^2(\ell + \kappa))$  bits for broadcasting an  $\ell$ -bit message (see Protocol ExBcast in Fig. 3 of section 2.2).

**Our Motivation and Results:** Given the importance of broadcast and the asynchronous communication setting, it is desirable to have a multi-valued broadcast protocol with better resilience. However to the best of our knowledge, there is no multi-valued asynchronous broadcast protocol with  $t \geq n/3$  and communication complexity  $\mathcal{O}(n\ell)$  bits. This work is the first attempt in this direction.

Assuming a set-up for implementing non-equivocation mechanism, we present a multi-valued asynchronous reliable broadcast protocol with  $t < n/2$ . Our multi-valued protocol allows a sender to reliably broadcast a message  $m$  of size  $\ell$  bits, with  $\ell \geq (t+1)\kappa$ , incurring a communication complexity of  $\mathcal{O}(n\ell) + \mathcal{BC}(\mathcal{O}(n\kappa))$  bits. Instantiating the broadcast (for shorter message) with the existing non-equivocation based broadcast protocol ExBcast (see Fig. 3 of section 2.2), our multi-valued broadcast protocol results in a total communication complexity of  $\mathcal{O}(n\ell + n^3\kappa)$  bits. So if  $\ell$  is sufficiently large, namely if  $\ell = \Omega(n^2\kappa)$ , then our protocol will have communication complexity  $\mathcal{O}(n\ell)$  bits, which is asymptotically optimal in  $\ell$ .

Even though our multi-valued broadcast protocol has non-optimal resilience<sup>2</sup>, it has optimal communication complexity (provided  $\ell$  is

<sup>1</sup>For example, consider a set of four parties  $P_1, P_2, P_3, P_4$  and say  $P_2$  is corrupted. If  $P_2$  is supposed to send some value  $x$  to all the remaining three parties and if  $P_2$  sends  $x$  using a non-equivocation mechanism only to  $P_1, P_3$ , then only  $P_1, P_3$  will output the same  $x$ , while  $P_4$  will not obtain  $x$ .

<sup>2</sup>The optimal resilience for non-equivocation based asynchronous

sufficiently large). Implementing the non-equivocation mechanism using a PKI set-up, along with a trusted hardware module (see section 2 on possible practical implementations of non-equivocation mechanism), we get the first multi-valued broadcast protocol in the asynchronous setting with communication complexity  $\mathcal{O}(n\ell)$  bits and with resilience  $t \geq n/3$ . Our protocol provides a trade-off between optimal resilience and optimal communication complexity. In applications where parties need to broadcast huge messages and where communication complexity is the key resource (such as MPC protocols), it is desirable to have a multi-valued broadcast protocol with better communication complexity. Our protocol fits the bill appropriately for such applications. Figure 1 gives a summary of the existing multi-valued broadcast protocols and our results.

**Open Problems:** Our protocol has non-optimal resilience. Designing a non-equivocation based multi-valued asynchronous broadcast protocol with  $t < n$  and with communication complexity  $\mathcal{O}(n\ell)$  is left as a challenging open problem.

**Other Related Works:** A lot of work has been done in the past to restrict the behaviour of Byzantine faults and prevent corrupted parties from sending conflicting information to different parties, both in the synchronous as well as asynchronous setting (see for example Chapter 12 of [1] and its references). These works try to simulate non-equivocation by introducing additional communication among the parties. However, the applicability of these techniques in the asynchronous setting and with  $t < n/2$  is not known and so we do not compare our result with these works.

## 2. PRELIMINARIES

We assume a set of  $n$  parties  $\mathcal{P} = \{P_1, \dots, P_n\}$ , connected by pair-wise authentic channels. There exists a computationally bounded adversary  $\text{Adv}$ , who can corrupt any  $t < n/2$  parties during the execution of a protocol and force the parties under its control (called corrupted parties) to behave in any arbitrary manner during the protocol execution. We assume the adversary is *static*, who decides the set of corrupted parties at the beginning of the execution of a protocol. For simplicity, we assume  $n = 2t + 1$ , so that  $t = \Theta(n)$ .

The communication channels among the parties are asynchronous allowing arbitrary, but a finite delay (i.e. the messages sent by the honest parties reach to their destinations eventually). The order of the message delivery is decided by a *scheduler*. To model the worst case scenario, we assume that the scheduler is under the control of the adversary. The scheduler can only schedule the messages exchanged between the honest parties, without having access to the “contents” of these messages. As in [5], we consider a protocol execution in this setting as a sequence of *atomic steps*, where a single party is *active* in each such step. A party is activated when it receives a message. On receiving a message, it performs an internal computation and then possibly sends messages on its outgoing channels. The order of the atomic steps are controlled by the scheduler. At the beginning of the computation, each party will be in a special *start* state. A party is said to *terminate/complete* the computation if it reaches a *halt* state, after which it does not perform any further computation. A protocol execution is said to be complete if all the honest parties terminate the computation. We assume that every message sent by a party during a protocol execution has a publicly known unique identifier (key) associated with it.

All computation in our protocols will be over a finite field  $\mathbb{F} = GF(2^\kappa)$ , where  $\kappa$  is the security parameter. Thus each element of

---

reliable broadcast is  $t < n$  [2, 15]

$\mathbb{F}$  can be represented by  $\mathcal{O}(\kappa)$  bits. We assume that  $\alpha_1, \dots, \alpha_n$  are distinct, non-zero publicly known elements of  $\mathbb{F}$ . The communication complexity of a protocol is defined to be the total number of bits communicated by the honest parties during the execution of the protocol. The communication complexity will have two components: communication which is done over the point-to-point channels and the communication which involves broadcasting some information to all the parties. We use the notation  $\mathcal{PC}(\cdot)$  and  $\mathcal{BC}(\cdot)$  respectively to express these two components of the communication complexity. We stress that while implementing any protocol, the broadcast communication involved in the protocol needs to be finally instantiated by running a reliable broadcast protocol over the point-to-point channels. The separation of the two components of communication complexity is just for simplicity.

A function  $\epsilon(\cdot) : \mathbb{N} \rightarrow \mathbb{R}^+$  is called *negligible* in  $\kappa$ , if for all  $c > 0$  there exists a  $\kappa_0$ , such that  $\epsilon(\kappa) < 1/\kappa^c$  for all  $\kappa > \kappa_0$ .

### 2.1 Definitions

We first define asynchronous reliable broadcast.

**DEFINITION 1 (Asynchronous Reliable Broadcast [9]).** Let  $\Pi$  be an asynchronous protocol executed among the set of parties  $\mathcal{P}$  and initiated by a special party called sender  $S \in \mathcal{P}$ , having input  $m$  (the message to be broadcast). Then  $\Pi$  is an asynchronous reliable broadcast protocol tolerating  $\text{Adv}$ , if the following holds, for every possible  $\text{Adv}$  and scheduler:

1. **Termination:** (a) If  $S$  is honest, then all honest parties eventually terminate  $\Pi$ . (b) If  $S$  is corrupted and some honest party terminates  $\Pi$ , then all honest parties eventually terminate  $\Pi$ .
2. **Correctness:** (a) If  $S$  is honest then every honest party outputs  $m$ . (b) If  $S$  is corrupted and some honest party outputs  $m'$ , then every honest party outputs  $m'$ .

We next discuss about non-equivocation mechanism.

**Transferable Non-equivocation:** The current discussion about non-equivocation mechanism is summarised from [2]; we refer the interested readers to [2] (and its references) for more details. A non-equivocation mechanism prevents a corrupted party from making conflicting statements to different parties. A transferable non-equivocation mechanism is similar to digital signatures. It allows a party to verifiably transfer a non-equivocation tag (or signature), provided by a party to other parties. A simplified and idealized definition for transferable non-equivocation is given in [2]. The simplified mechanism  $\text{Neq}$  (given in Fig. 2) models an instantiation of transferable non-equivocation and is parametrized by a security parameter  $\kappa$ . The mechanism has the following three phases:

- During the setup phase, each  $P_i$  gets associated with a unique non-equivocation list  $L_i$  (initially empty) and every party is informed about this association.
- The designated list owner party can create a non-equivocation signature for any key-message pair such that it cannot equivocate and obtain a signature for the same key twice, except with a negligible probability in  $\kappa$ .
- Given a key-message-signature triplet associated with a designated sender, any party successfully verifies only correctly generated signatures, except with a negligible probability in  $\kappa$ .

**Figure 1: Summary of the various multi-valued broadcast protocols for broadcasting  $\ell$ -bit message**

Network setting	Security	Corruption threshold	Communication complexity	Remark
Synchronous	Perfect	$t < n/3$	$\mathcal{O}(n\ell + \sqrt{\ell}n^4 + n^6)$ [32]	If $\ell = \Omega(n^6)$ then communication complexity is $\mathcal{O}(n\ell)$
Synchronous	Perfect	$t < n/3$	$\mathcal{O}(n\ell + n^4)$ [34]	If $\ell = \Omega(n^3)$ then communication complexity is $\mathcal{O}(n\ell)$
Synchronous	information-theoretic	$t < n/2$	$\mathcal{O}(n\ell + n^7\kappa)$ [22]	If $\ell = \Omega(n^6\kappa)$ then communication complexity is $\mathcal{O}(n\ell)$
Synchronous	Cryptographic	$t < n/2$	$\mathcal{O}(n\ell + n^4(n + \kappa))$ [22]	If $\ell = \Omega(n^3(n + \kappa))$ then communication complexity is $\mathcal{O}(n\ell)$
Synchronous	Information-theoretic	$t < n$	$\mathcal{O}(n\ell + n^{10}\kappa)$ [27]	If $\ell = \Omega(n^9\kappa)$ then communication complexity is $\mathcal{O}(n\ell)$
Synchronous	Cryptographic	$t < n$	$\mathcal{O}(n\ell + n^5\kappa)$ [27]	If $\ell = \Omega(n^4\kappa)$ then communication complexity is $\mathcal{O}(n\ell)$
Asynchronous	Perfect	$t < n/3$	$\mathcal{O}(n\ell + n^4 \log n)$ [34]	If $\ell = \Omega(n^3 \log n)$ then communication complexity is $\mathcal{O}(n\ell)$
Asynchronous	Cryptographic <sup>a</sup>	$t < n$	$\mathcal{O}(n^2(\ell + \kappa))$ [15,17]	<b>Non-optimal</b> communication complexity (in $\ell$ )
Asynchronous	Cryptographic <sup>a</sup>	$t < n/2$	$\mathcal{O}(n\ell + n^3\kappa)$ ( <b>this paper</b> )	If $\ell = \Omega(n^2\kappa)$ then communication complexity is $\mathcal{O}(n\ell)$

<sup>a</sup> Apart from pair-wise authentic channels, this protocol also requires a set-up for implementing non-equivocation mechanism, which can be done using PKI and a trusted hardware set-up (see section 2).

In [2], the authors survey the existing instantiations of transferable non-equivocation mechanism and analyze their relations to Neq. Most instantiations of transferable non-equivocation deploy an increment-only counter for keys and signatures with either a public key infrastructure (PKI) [14, 15, 17, 31] or message authentication codes (MACs) generated with a replicated secret key [29]. In Neq, these are generalized using the list  $L_i$  of key-message-signature triplets associated with  $P_i$ , indexed by *party-defined ordered keys*  $\mathbf{l}$ . Similar to signatures, only  $P_i$  can use NeqSign to add triplets (one per each key) to  $L_i$ . Similar to PKI, anybody can use NeqVerify to verify if a triplet  $(\mathbf{l}, m, \sigma)$  belongs to the list  $L_i$  of  $P_i$ , and can verifiably transfer authentication to others. The increment-only counter provides an efficient way to implement  $L_i$  as only the counter value has to be maintained, instead of the whole list.

In the rest of the paper, we assume that each non-equivocation signature/tag has size  $\mathcal{O}(\kappa)$  bits. Let  $P_D \in \mathcal{P}$  be a designated party and let  $P_i, P_j \in \mathcal{P}$ . We will say that  $P_D$  sends  $\{m\}_{\sigma_D}$  to  $P_i$  to mean that  $P_D$  sends a triplet  $(\mathbf{l}, m, \sigma_D^{\mathbf{l}, m})$  for a pre-defined key  $\mathbf{l}$  to  $P_i$ . The term  $P_i$  receives  $\{m\}_{\sigma_D}$  from  $P_D$  will mean that  $P_i$  outputs  $m$  after applying NeqVerify to check if  $\sigma_D^{\mathbf{l}, m}$  is obtained by  $P_D$  using NeqSign on  $\mathbf{l}$  and  $m$ . Similarly we will say that  $P_i$  forwards  $\{m\}_{\sigma_D}$  to  $P_j$  to denote that  $P_i$  sends the message  $\{m\}_{\sigma_D}$  (on the behalf of  $P_D$ ) to  $P_j$ , who subsequently receives the message (on the behalf of  $P_D$ ) in the above sense. The keys  $\mathbf{l}$  are avoided in above phrases for simplicity, as they can be easily pre-assigned to protocol instance-step combinations in an unambiguous manner.

## 2.2 Existing Reliable Broadcast Protocol Based on Non-equivocation

Assuming a transferable non-equivocation mechanism, an asynchronous reliable broadcast protocol with  $t < n$  was presented in [15,17]. The protocol has communication complexity  $\mathcal{PC}(\mathcal{O}(n^2(\ell + \kappa)))$  bits for broadcasting an  $\ell$ -bit message. The high-level idea of the protocol is as follows: to broadcast a message  $m$ , the sender  $S$  first (non-equivocally) sends  $m$  to all the parties; this prevents

a corrupted  $S$  from sending different messages to different (honest) parties. However, a corrupted  $S$  can avoid sending  $m$  to some honest parties. To get rid of this problem, as soon as a party non-equivocally receives  $m$  from  $S$ , it non-equivocally forwards it to every other party on the behalf of  $S$ . This ensures that if some honest party receives  $m$ , then every other honest party also eventually receives  $m$ . For the sake of completeness, the non-equivocation based broadcast protocol of [15, 17] is recalled from [2] and presented in Fig. 3.

The properties of ExBcast are stated in Lemma 1; for the proof we refer to [2, 15, 17]. In the rest of the paper, the term “ $P_i$  broadcasts  $m$ ” means that  $P_i$  as a sender  $S$  invokes an instance of ExBcast for broadcasting  $m$ . Similarly, “ $P_j$  receives  $m$  from the broadcast of  $P_i$ ” means that  $P_j$  terminates the instance of ExBcast invoked by  $P_i$  with output  $m$ .

**LEMMA 1.** *Let  $S \in \mathcal{P}$  has input  $m$  for protocol ExBcast, where  $|m| = \ell$  bits. Assuming a non-equivocation mechanism, protocol ExBcast achieves the following for every possible Adv and every possible scheduler:*

- **Termination:**

- If  $S$  is honest then except with a negligible probability (in  $\kappa$ ), every honest party eventually terminates the protocol.
- If  $S$  is corrupted and some honest party terminates the protocol, then except with a negligible probability (in  $\kappa$ ), every other honest party eventually terminates the protocol .

- **Correctness:**

- If  $S$  is honest then except with a negligible probability (in  $\kappa$ ), all honest parties output  $m$ .
- If  $S$  is corrupted and some honest party outputs  $m'$ , then except with a negligible probability (in  $\kappa$ ), all honest parties output  $m'$ .

**Figure 2: A simplified transferable non-equivocation mechanism [2]**

The Neq Mechanism
Neq is parameterized by a polynomial $p(\cdot)$ , and an implicit security parameter $\kappa$ .
• <b>Set-up:</b> Upon receiving a $(\text{Setup})$ message from party $P_i \in \mathcal{P}$ , do:
1. If a list $L_i$ exists, return $\perp$ .
2. Otherwise, create an empty list $L_i$ of key-message-signature triplets of type $\{0, 1\}^* \times \{0, 1\}^* \times \{0, 1\}^{p(\kappa)}$ . Send a message $(\text{Registered}, P_i)$ to all parties in $\mathcal{P}$ .
• <b>Signing:</b> Upon receiving an $(\text{NeqSign}, P_i, l, m)$ message from party $P_i$ , do:
1. If the list $L_i$ does not exist or $(l, \star, \star) \in L_i$ , return $\perp$ .
2. Otherwise, choose an arbitrary (signature) tag $\sigma_i^{l,m} \in \{0, 1\}^{p(\kappa)}$ , update $L_i \leftarrow L_i \cup (l, m, \sigma_i^{l,m})$ and return $\sigma_i^{l,m}$ .
• <b>Verification:</b> Upon receiving an $(\text{NeqVerify}, P_j, l, m, \sigma)$ message from party $P_j \in \mathcal{P}$ , do:
1. If the list $L_i$ does not exist or $(l, m, \sigma) \notin L_i$ , then return 0, else return 1.

**Figure 3: The existing non-equivocation based broadcast protocol**

$\text{ExBcast}(S, m, \mathcal{P}) :  m  = \ell$
<b>Distributing the message</b> — Only S executes the following code:
• Send $\{m\}_{\sigma_S}$ to every party $P_i \in \mathcal{P}$ .
<b>Termination</b> — Every party $P_i$ , including S executes the following code:
• If $\{m\}_{\sigma_S}$ is received from S, then forward $\{m\}_{\sigma_S}$ to every $P_j \in \mathcal{P}$ , output $m$ and terminate.
• If $\{m\}_{\sigma_S}$ is forwarded by some party in $\mathcal{P}$ , then forward $\{m\}_{\sigma_S}$ to every $P_j \in \mathcal{P}$ , output $m$ and terminate.

- **Communication Complexity:** *The protocol has communication complexity  $\mathcal{O}(n^2(\ell + \kappa))$  bits.*

### 3. MULTI-VALUED ASYNCHRONOUS BROADCAST

For the ease of understanding, we first present a multi-valued broadcast protocol NeqBcast, which does not have communication complexity  $\mathcal{O}(n\ell)$  bits. We will then show how a slight modification of NeqBcast leads to our multi-valued broadcast protocol NeqBcastOptimal with communication complexity  $\mathcal{O}(n\ell)$ .

#### 3.1 Protocol NeqBcast

Let  $m$  be the message which S wants to broadcast to the  $n$  parties, such that  $|m| = (t+1)\kappa = \Theta(n\kappa)$  bits<sup>3</sup>. We present a protocol NeqBcast, which allows S to broadcast the message with communication complexity  $\mathcal{PC}(\mathcal{O}(n^2\kappa))$  bits and  $\mathcal{BC}(\mathcal{O}(n\kappa))$  bits. The protocol is given in Fig. 4. The high level idea of the protocol is as follows: S first “encodes” the message as a polynomial of degree at most  $t$  over  $\mathbb{F}$ . Namely, each block of  $\kappa$  bits of  $m$  is represented as an element of  $\mathbb{F}$  and a polynomial  $M(x)$  is formed using the resultant elements of  $\mathbb{F}$  as coefficients. To enable every (honest) party to obtain  $m$ , the goal is to ensure that every (honest) party eventually receives  $M(x)$ . Once this is done then  $m$  can be recovered by converting back each coefficient of  $M(x)$  into its  $\kappa$ -bit representation and concatenating all those blocks of  $\kappa$  bits.

The naive way to ensure that every party receives  $M(x)$  is by using a similar idea used in the existing non-equivocation based broadcast protocol ExBcast: S can first (non-equivocally) send  $M(x)$  (by sending its coefficients) to every party. And any party on

receiving  $M(x)$ , non-equivocally forwards it to every other party on behalf of S. Unfortunately the communication complexity of this naive broadcast protocol will be  $\mathcal{O}(\mathcal{PC}(n^3\kappa))$  bits, while our goal is to restrict the point-to-point communication to  $\mathcal{O}(\mathcal{PC}(n^2\kappa))$ . So we follow a slightly different approach, which is a reminiscent of Shamir secret-sharing [38]. Instead of sending  $M(x)$  to the parties by sending its coefficients, S computes  $n$  distinct evaluations of  $M(x)$  and the goal now is to ensure that each (honest) party correctly obtains  $t+1$  of these evaluations, which are enough to interpolate  $M(x)$ . For this S first (non-equivocally) sends these  $n$  evaluations to all the parties. If S is *honest* then every honest party eventually receives these evaluations. However if S is *corrupted*, then it may not send the evaluations to *all* the honest parties. To get rid of this problem, we ask the parties to (non-equivocally) forward the evaluations upon receiving them from S. However to prevent high communication complexity, the parties are asked to forward *only* designated evaluations to designated parties. In a more detail, the parties first try to find out and agree upon a set of  $t+1$  parties that have received the full list of  $n$  evaluations. This is done by asking each party to broadcast a signal after receiving (and verifying) the full list of  $n$  evaluations and then by further asking S to broadcast a list of  $t+1$  signallers. Call this set of  $t+1$  parties as CORE. The deployment of non-equivocation mechanism ensures that every honest party in CORE has received the same set of  $n$  evaluations of a unique polynomial of degree at most  $t$ . Moreover, it is guaranteed that there is at least one honest party in CORE.

Once CORE is found, we add the following two “rounds” of relay communication, which ensure that every honest party eventually receives  $t+1$  correct evaluations. First we ask the parties in CORE to (non-equivocally) forward only the  $i$ th evaluation to party  $P_i$  on the behalf of S. This ensures that every honest party  $P_i$  outside CORE receives the  $i$ th evaluation of  $M(x)$ . The deploy-

<sup>3</sup>Recall that for  $n = 2t + 1$ , we have  $t = \Theta(n)$ .

ment of non-equivocation mechanism prevents a corrupted party in CORE from forwarding an incorrect evaluation to any party outside CORE. At the end of the first round of relay communication, *every* honest party  $P_j$  holds the  $j$ th evaluation. During the second round of the relay communication, each party  $P_j$  forwards the  $j$ th evaluation to every other party. The evaluation of every honest party eventually reaches every other honest party and since there are at least  $t + 1$  honest parties, each party eventually receives sufficient number of evaluations to reconstruct  $M(x)$ . Again the deployment of the non-equivocation mechanism prevents a corrupted party from forwarding an incorrect evaluation to any honest party. The way two rounds of relay communication is structured, it is ensured that the communication over the point-to-point channels in the protocol is  $\mathcal{PC}(\mathcal{O}(n^2\kappa))$  bits. In the protocol, the instances of broadcast are invoked for broadcasting the identity of CORE and for broadcasting the signals and hence it is independent of the actual message, which needs to be broadcasted by S.

**LEMMA 2 (Termination).** *For every possible Adv and for every possible scheduler, protocol NeqBcast achieves the following:*

- *If S is honest then except with a negligible probability in  $\kappa$ , every honest party eventually terminates the protocol.*
- *If S is corrupted and some honest party terminates the protocol, then except with a negligible probability in  $\kappa$  every other honest party eventually terminates the protocol.*

**Proof:** If S is *honest*, then every honest party eventually receives  $\{m_1\}_{\sigma_S}, \dots, \{m_n\}_{\sigma_S}$  from S. Moreover,  $m_1, \dots, m_n$  will lie on a polynomial  $M(x)$  of degree at most  $t$ . Hence each honest party eventually broadcasts the  $(\text{OK}, *)$  message. As there are at least  $t + 1$  honest parties who eventually broadcasts  $(\text{OK}, *)$ , S eventually finds a CORE and broadcasts the same. So every honest party eventually receives CORE and verifies the same (this follows from the property of broadcast). This is followed by every honest party  $P_j$  in CORE forwarding the  $i$ th share to every party  $P_i$  outside CORE. Since CORE will have at least one honest party, it is ensured that every honest party  $P_i$  outside CORE eventually receives  $m_i$ , which it later forwards to every other party  $P_j$ . It thus follows that every honest party  $P_j$  eventually receives  $t + 1$   $m_i$  values, using which it reconstructs the  $M(x)$  polynomial and hence the message m and terminate.

Now consider the case when S is *corrupted* and some honest party, say  $P_h \in \mathcal{P}$ , terminates the protocol. Since  $P_h$  terminates the protocol, it implies that it received a CORE with  $|\text{CORE}| \geq t + 1$  from the broadcast of S. By the property of broadcast, every other honest party also eventually does the same. Termination of the protocol by  $P_h$  further implies that  $P_h$  received  $(\text{OK}, *)$  message from the broadcast of every party in CORE and the properties of broadcast imply that every honest party eventually receives the same. Since  $|\text{CORE}| \geq t + 1$ , it implies that it has at least one honest party. The termination property now follows the same argument as used for the case of honest S.

**LEMMA 3 (Correctness).** *For every possible Adv and for every possible scheduler, protocol NeqBcast achieves the following, except with a negligible probability in  $\kappa$ :*

- *If S is honest then every honest party outputs m.*
- *If S is corrupted and some honest party outputs  $m'$ , then every other honest party outputs  $m'$ .*

**Proof:** We first consider the case when S is *honest*. Let  $P_h$  be an arbitrary *honest* party. We show that  $P_h$  outputs m (with high

probability). As argued in the proof of the termination property for an honest S, every honest party eventually receives a valid CORE with  $|\text{CORE}| \geq t + 1$  from the broadcast of S. Moreover there will be at least one honest party in CORE, who received  $n$  distinct values of  $M(x)$  and forwarded the  $i$ th value  $m_i$  of  $M(x)$  to every party  $P_i$  outside CORE. This implies that each honest  $P_i$  eventually possess  $m_i$ , which is forwarded by  $P_i$  to every other party. This further implies that  $P_h$  eventually receives  $t + 1$  distinct values of  $M(x)$ , which are sufficient to recover  $M(x)$ . To complete the proof, it is enough to argue that the  $t + 1$   $m_i$  values used by  $P_h$  to recover the polynomial are indeed correct. For this we consider two possible cases. If an  $m_i$  value is forwarded by an honest  $P_i$ , then  $m_i$  is indeed a correct value. On the other hand if  $P_i$  is corrupted, then from the properties of non-equivocation mechanism it follows that  $m_i$  is indeed a correct value with high probability. This is because with high probability, a corrupted  $P_i$  will fail to generate S's non-equivocation signature on an incorrect  $m_i$ .

Next consider the case when S is *corrupted* and some honest party, say  $P_h$ , outputs  $m'$ . We show that every other honest party different from  $P_h$ , say  $P_\alpha$ , also outputs  $m'$ . Since  $P_h$  outputs  $m'$ , it implies that it has verified the existence of a valid CORE, with  $|\text{CORE}| \geq t + 1$  and containing at least one honest party. Each honest party in CORE would have received  $\{m_1\}_{\sigma_S}, \dots, \{m_n\}_{\sigma_S}$  from S and verified that  $(\alpha_1, m_1), \dots, (\alpha_n, m_n)$  lie on a unique polynomial of degree at most  $t$ , say  $M'(x)$ . We let  $m'$  to be the message of size  $(t + 1)\kappa$  bits, defined by the  $t + 1$  coefficients of  $M'(x)$ . From the properties of non-equivocation mechanism, it follow that with high probability, each honest party in CORE would receive the same  $\{m_1\}_{\sigma_S}, \dots, \{m_n\}_{\sigma_S}$  from S. We next claim that each honest  $P_i$  outside CORE eventually possess  $m_i$ , the  $i$ th value of  $M'(x)$ . For this we note that  $m_i$  is forwarded to such parties  $P_i$  by some party in CORE. If the forwarding party is an honest party from CORE, then the claim is trivially true. However, the claim is true even if the forwarding party is a corrupted party from CORE. This is because a corrupted  $P_j \in \text{CORE}$  cannot create S's non-equivocation signature on an incorrect  $m_i$  and forward it to  $P_i$ . To complete the proof, we finally argue that the  $t + 1$  forwarded values that  $P_\alpha$  utilizes to recover the message are indeed the values of  $M'(x)$ . This trivially holds for the values forwarded by the honest parties. The properties of non-equivocation ensure that the same is true even for the values forwarded by the corrupted parties. This is because a corrupted party will fail to generate S's non-equivocation signature on an incorrect  $m_i$  value and forward it to  $P_\alpha$ .

**LEMMA 4 (Communication Complexity).** *Protocol NeqBcast has communication complexity  $\mathcal{PC}(\mathcal{O}(n^2\kappa))$  bits and  $\mathcal{BC}(\mathcal{O}(n\kappa))$  bits.*

**Proof:** In the protocol, S has to non-equivocally send a vector of  $n$  values of  $M(x)$  to each party, which costs  $\mathcal{PC}(\mathcal{O}(n^2\kappa))$  bits. In addition, each party in CORE has to non-equivocally forward one value of  $M(x)$  to every party outside CORE. In total there can be  $\mathcal{O}(n^2)$  such pair of parties and so this step has total communication complexity  $\mathcal{PC}(\mathcal{O}(n^2\kappa))$  bits. Finally to reconstruct the  $M(x)$  polynomial, each party forwards its  $m_i$  value to every other party, which costs  $\mathcal{PC}(\mathcal{O}(n^2\kappa))$  bits. Each party may broadcast an OK signal and S may broadcast a list of  $\mathcal{O}(n)$  parties, constituting the CORE. So the broadcast communication complexity of the protocol is  $\mathcal{BC}(\mathcal{O}(n\kappa))$  bits.

The following theorem now trivially follows from Lemma 2-4.

**THEOREM 2.** *Let S  $\in \mathcal{P}$  has input m for NeqBcast, where  $|m| = (t + 1)\kappa = \Theta(n\kappa)$  bits. Assuming a set-up for non-equivocation mechanism for the n parties, for every possible Adv and scheduler, protocol NeqBcast is a reliable broadcast protocol,*

**Figure 4: Protocol NeqBcast**

$\text{NeqBcast}(\mathcal{S}, m, \mathcal{P}) : |m| = (t + 1)\kappa$  bits

**Distributing the message** — Only  $\mathcal{S}$  executes the following code:

- Divide the message  $m$  into blocks of  $\kappa$  bits each and interpret each block of  $\kappa$  bits as an element of  $\mathbb{F}$ . Let  $a_0, \dots, a_t$  be the resultant field elements.
- Define  $M(x) \stackrel{\text{def}}{=} a_0 + a_1x + \dots + a_tx^t$ . Compute  $m_i \stackrel{\text{def}}{=} M(\alpha_i)$ , for  $i = 1, \dots, n$ .
- Send  $\{m_1\}_{\sigma_S}, \dots, \{m_n\}_{\sigma_S}$  to each  $P_i \in \mathcal{P}$ .

**Deciding the CORE** — Every party  $P_i$ , including  $\mathcal{S}$  executes the following code:

- On receiving  $\{m_1\}_{\sigma_S}, \dots, \{m_n\}_{\sigma_S}$  from  $\mathcal{S}$ , verify if  $(\alpha_1, m_1), \dots, (\alpha_n, m_n)$  lie on a unique polynomial of degree at most  $t$  over  $\mathbb{F}$ . If yes, then broadcast  $(\text{OK}, P_i)$  message.
- If  $P_i = \mathcal{S}$ , then additionally it does the following:
  - Create an accumulative set  $\text{CORE}$ , initialized to  $\emptyset$ . Include party  $P_i$  to  $\text{CORE}$ , if  $(\text{OK}, P_i)$  is received from the broadcast of  $P_i$ .
  - Wait till  $|\text{CORE}| \geq t + 1$ . Once  $|\text{CORE}| \geq t + 1$ , broadcast  $\text{CORE}$ .

**Verifying CORE and Relaying Shares of the  $M(x)$  Polynomial** — Every party  $P_j$ , including  $\mathcal{S}$  executes the following code:

- Wait to receive a set  $\text{CORE}$  with  $|\text{CORE}| \geq t + 1$  from the broadcast of  $\mathcal{S}$ .
- On receiving a  $\text{CORE}$  with  $|\text{CORE}| \geq t + 1$  from the broadcast of  $\mathcal{S}$ , wait to receive  $(\text{OK}, P_i)$  from the broadcast of every  $P_i \in \text{CORE}$ . On receiving, do the following:
  - If  $P_j \in \text{CORE}$ , then for every  $P_i \notin \text{CORE}$ , forward  $\{m_i\}_{\sigma_S}$  to party  $P_i$ .
  - If  $P_j \notin \text{CORE}$ , then wait for  $\{m_j\}_{\sigma_S}$  to be forwarded by some party  $P_i \in \text{CORE}$ .

**Termination** — Every party  $P_j$ , including  $\mathcal{S}$  executes the following code:

- Forward  $\{m_j\}_{\sigma_S}$  to every  $P_i \in \mathcal{P}$ .
- Wait for  $\{m_i\}_{\sigma_S}$  values to be forwarded by at least  $t + 1$  parties  $P_i$ . On receiving, interpolate a polynomial  $M'(x)$  of degree at most  $t$  using the received  $m_i$  values.
- Let  $(a'_0, \dots, a'_t)$  be the coefficients of  $M'(x)$ . Interpret each of these coefficients as a block of  $\kappa$  bits and let  $m'$  be the concatenation of all such blocks of  $\kappa$  bits. Output  $m'$  and terminate.

which satisfies Definition 1, except with a negligible probability in  $\kappa$ . The protocol has communication complexity  $\mathcal{PC}(\mathcal{O}(n^2\kappa))$  bits and  $\mathcal{BC}(\mathcal{O}(n\kappa))$  bits.

### 3.2 Modifying NeqBcast to Achieve Optimal Communication Complexity

We now show how to modify NeqBcast to get a multi-valued reliable broadcast protocol with optimal communication complexity of  $\mathcal{O}(n\ell)$  bits. Let  $m$  be the message which  $\mathcal{S}$  wants to broadcast, such that  $|m| = \ell$  bits, where  $\ell \geq (t + 1)\kappa$ . We present a reliable broadcast protocol NeqBcastOptimal, which allows  $\mathcal{S}$  to broadcast the message with communication complexity  $\mathcal{PC}(\mathcal{O}(n\ell))$  and  $\mathcal{BC}(\mathcal{O}(n\kappa))$  bits. Instantiating the broadcast with non-equivocation based broadcast protocol ExBcast of [2], NeqBcastOptimal results in a communication complexity of  $\mathcal{PC}(\mathcal{O}(n\ell + n^3\kappa))$  bits. So if  $\ell$  is sufficiently large, namely if  $\ell = \Omega(n^2\kappa)$ , then NeqBcastOptimal will have communication complexity  $\mathcal{PC}(\mathcal{O}(n\ell))$  bits.

The naive way of designing NeqBcastOptimal using NeqBcast is to divide  $m$  into sub-blocks of size  $(t + 1)\kappa$  bits and use a dedicated instance of NeqBcast for broadcasting each sub-block. This will have communication complexity  $\mathcal{PC}(\mathcal{O}(\frac{\ell}{(t+1)\kappa}n^2\kappa)) = \mathcal{PC}(\mathcal{O}(n\ell))$  and<sup>4</sup>  $\mathcal{BC}(\mathcal{O}(\frac{\ell}{(t+1)\kappa}n\kappa)) = \mathcal{BC}(\ell)$ . So the broadcast communication complexity will be *dependent* on  $\ell$ . To ensure that the broadcast communication complexity remains independent of  $\ell$ , we ensure that each party broadcasts only once on the behalf of all the instances of NeqBcast that are invoked. Namely, a party broadcasts an  $(\text{OK}, \star)$  message only if the “pre-condition” for broadcast

ing the  $(\text{OK}, \star)$  message is satisfied in *all* the instances of NeqBcast that are invoked. Specifically, each party  $P_i$  now broadcasts an  $(\text{OK}, P_i)$  message only after verifying that the set of  $n$  values received in various instances of NeqBcast lie on polynomials of degree at most  $t$ . In the same way,  $\mathcal{S}$  creates only one  $\text{CORE}$  on the behalf of all underlying instances of NeqBcast and broadcasts the same. It is easy to see that now the resultant protocol will have broadcast communication complexity  $\mathcal{BC}(\mathcal{O}(n\kappa))$  bits. Protocol NeqBcastOptimal is formally presented in Fig. 5.

The following theorem follows in a straight forward fashion from the above discussion and Theorem 2

**THEOREM 3.** *Let  $\mathcal{S} \in \mathcal{P}$  has input  $m$  for NeqBcastOptimal, where  $|m| = \ell$  bits, with  $\ell \geq (t + 1)\kappa$ . Assuming a set-up for non-equivocation mechanism for the  $n$  parties, for every possible Adv and scheduler, protocol NeqBcastOptimal is a reliable broadcast protocol which satisfies Definition 1, except with a negligible probability in  $\kappa$ . The protocol has communication complexity  $\mathcal{PC}(\mathcal{O}(n\ell))$  bits and  $\mathcal{BC}(\mathcal{O}(n\kappa))$  bits.*

## 4. REFERENCES

- [1] Hagit Attiya and Jennifer Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics* (2nd edition). John Wiley Interscience, March 2004.
- [2] M. Backes, F. Bendun, A. Choudhury, and A. Kate. Asynchronous MPC with a Strict Honest Majority Using Non-equivocation. In *PODC*, pages 10–19. ACM, 2014.
- [3] Z. Beerliová-Trubíniová and M. Hirt. Efficient multi-party computation with dispute control. In S. Halevi and T. Rabin,

<sup>4</sup>Recall that for  $n = 2t + 1$ , we have  $t = \Theta(n)$ .

**Figure 5: Protocol NeqBcastOptimal**

**NeqBcastOptimal( $S, m, \mathcal{P}$ )** :  $|m| = \ell, \ell \geq (t + 1)\kappa$

**Distributing the message** — Only  $S$  executes the following code:

- Divide the message  $m$  into sub-blocks of  $(t + 1)\kappa$  bits each. Let  $m^{(1)}, \dots, m^{(B)}$  be the sub-blocks of  $m$ , each of size  $(t + 1)\kappa$  bits, where  $B \stackrel{\text{def}}{=} \frac{\ell}{t+1}$ .
- For  $l = 1, \dots, B$ , do the following:
  - Divide  $m^{(l)}$  into  $t + 1$  blocks of  $\kappa$  bits and interpret each block of  $\kappa$  bits as an element of  $\mathbb{F}$ . Let  $a_0^{(l)}, \dots, a_t^{(l)}$  be the resultant field elements.
  - Define  $M^{(l)}(x) \stackrel{\text{def}}{=} a_0^{(l)} + a_1^{(l)}x + \dots + a_t^{(l)}x^t$ . Compute  $m_i^{(l)} \stackrel{\text{def}}{=} M^{(l)}(\alpha_i)$ , for  $i = 1, \dots, n$ .
  - Send  $\{m_1^{(l)}\}_{\sigma_S}, \dots, \{m_n^{(l)}\}_{\sigma_S}$  to each  $P_i \in \mathcal{P}$ .

**Deciding the CORE** — Every party  $P_i$ , including  $S$  executes the following code:

- For  $l = 1, \dots, B$ , on receiving  $\{m_1^{(l)}\}_{\sigma_S}, \dots, \{m_n^{(l)}\}_{\sigma_S}$  from  $S$ , verify if  $(\alpha_1, m_1^{(l)}), \dots, (\alpha_n, m_n^{(l)})$  lie on a polynomial of degree at most  $t$  over  $\mathbb{F}$ . If the verification is successful for every  $l = 1, \dots, B$ , then broadcast  $(\text{OK}, P_i)$  message.
- If  $P_i = S$ , then additionally it does the following:
  - Create an accumulative set  $\text{CORE}$ , initialized to  $\emptyset$ . Include party  $P_i$  to  $\text{CORE}$ , if  $(\text{OK}, P_i)$  is received from the broadcast of  $P_i$ .
  - Wait till  $|\text{CORE}| \geq t + 1$ . Once  $|\text{CORE}| \geq t + 1$ , broadcast  $\text{CORE}$ .

**Verifying CORE and Relaying Shares of the  $M(x)$  Polynomials** — Every party  $P_j$ , including  $S$  executes the following code:

- Wait to receive a set  $\text{CORE}$  with  $|\text{CORE}| \geq t + 1$  from the broadcast of  $S$ .
- On receiving a  $\text{CORE}$  with  $|\text{CORE}| \geq t + 1$  from the broadcast of  $S$ , wait to receive  $(\text{OK}, P_i)$  from the broadcast of every  $P_i \in \text{CORE}$ . On receiving, do the following:
  - If  $P_j \in \text{CORE}$ , then for every  $P_i \notin \text{CORE}$ , forward  $\{m_i^{(1)}\}_{\sigma_S}, \dots, \{m_i^{(B)}\}_{\sigma_S}$  to party  $P_i$ .
  - If  $P_j \notin \text{CORE}$ , then wait for  $\{m_j^{(1)}\}_{\sigma_S}, \dots, \{m_j^{(B)}\}_{\sigma_S}$  to be forwarded by some party  $P_i \in \text{CORE}$ .

**Termination** — Every party  $P_j$ , including  $S$  executes the following code:

- Forward  $\{m_j^{(1)}\}_{\sigma_S}, \dots, \{m_j^{(B)}\}_{\sigma_S}$  to every  $P_i \in \mathcal{P}$ .
- For  $l = 1, \dots, B$ , do the following:
  - Wait for  $\{m_i^{(l)}\}_{\sigma_S}$  values to be forwarded by at least  $t + 1$  parties  $P_i$ . On receiving, interpolate a polynomial  $M'^{(l)}(x)$  of degree at most  $t$  using the received  $m_i^{(l)}$  values.
  - Let  $(a_0'^{(l)}, \dots, a_t'^{(l)})$  be the coefficients of  $M'^{(l)}(x)$ . Interpret each of these coefficients as a block of  $\kappa$  bits and let  $m'^{(l)}$  be the concatenation of all such blocks of  $\kappa$  bits.
- Let  $m'$  be the concatenation of  $m'^{(1)}, \dots, m'^{(B)}$ . Output  $m'$  and terminate.

- editors, *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*, pages 305–328. Springer Verlag, 2006.
- [4] Z. Beerliov-Trubniov and M. Hirt. Perfectly-Secure MPC with Linear Communication Complexity. In R. Canetti, editor, *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*, volume 4948 of *Lecture Notes in Computer Science*, pages 213–230. Springer Verlag, 2008.
- [5] M. Ben-Or, R. Canetti, and O. Goldreich. Asynchronous secure computation. In S. R. Kosaraju, D. S. Johnson, and A. Aggarwal, editors, *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 52–61. ACM, 1993.
- [6] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract). In J. Simon, editor,

*Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10. ACM, 1988.

- [7] P. Berman, J. A. Garay, and K. J. Perry. Towards Optimal Distributed Consensus (Extended Abstract). In *FOCS*, pages 410–415. IEEE Computer Society, 1989.
- [8] G. Bracha. An Asynchronous [(n-1)/3]-Resilient Consensus Protocol. In T. Kameda, J. Misra, J. Peters, and N. Santoro, editors, *Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing, Vancouver, B. C., Canada, August 27-29, 1984*, pages 154–162. ACM, 1984.
- [9] R. Canetti. *Studies in Secure Multiparty Computation and Applications*. PhD thesis, Weizmann Institute, Israel, 1995.
- [10] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults (Extended Abstract). In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 383–395. IEEE Computer Society, 1985.

- [11] A. Choudhury, E. Orsini, A. Patra, and N. P. Smart. Linear Overhead Robust MPC with Honest Majority Using Preprocessing. *IACR Cryptology ePrint Archive*, 2015:705, 2015. Preliminary version accepted in SCN 2016.
- [12] A. Choudhury and A. Patra. Optimally Resilient Asynchronous MPC with Linear Communication Complexity. In *ICDCN*, pages 5:1–5:10. ACM, 2015.
- [13] A. Choudhury, A. Patra, and N. P. Smart. Reducing the Overhead of Cloud MPC. *IACR Cryptology ePrint Archive*, 2014:105, 2014. Preliminary version accepted in SCN 2014.
- [14] B. Chun, P. Maniatis, S. Shenker, and J. Kubiatowicz. Attested Append-only Memory: Making Adversaries Stick to their Word. In *SOSP*, pages 189–204. ACM, 2007.
- [15] A. Clement, F. Junqueira, A. Kate, and R. Rodrigues. On the (limited) Power of Non-equivocation. In *PODC*, pages 301–308. ACM, 2012.
- [16] B. A. Coan and J. L. Welch. Modular Construction of a Byzantine Agreement Protocol with Optimal Message Bit Complexity. *Inf. Comput.*, 97(1):61–85, 1992.
- [17] M. Correia, G. S. Veronese, and L. C. Lung. Asynchronous Byzantine consensus with  $2f+1$  Processes. In *SAC*, pages 475–480. ACM, 2010.
- [18] I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias. Multiparty Computation from Somewhat Homomorphic Encryption. In R. Safavi-Naini and R. Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662. Springer, 2012.
- [19] D. Dolev and R. Reischuk. Bounds on Information Exchange for Byzantine Agreement. *J. ACM*, 32(1):191–204, 1985.
- [20] D. Dolev and H. R. Strong. Authenticated Algorithms for Byzantine Agreement. *SIAM J. Comput.*, 12(4):656–666, 1983.
- [21] M. Fitzi, J. A. Garay, S. Gollakota, C. Pandu Rangan, and K. Srinathan. Round-Optimal and Efficient Verifiable Secret Sharing. In S. Halevi and T. Rabin, editors, *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*, pages 329–342. Springer, 2006.
- [22] M. Fitzi and M. Hirt. Optimally Efficient Multi-valued Byzantine Agreement. In E. Ruppert and D. Malkhi, editors, *PODC*, pages 163–168. ACM Press, 2006.
- [23] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. The Round Complexity of Verifiable Secret Sharing and Secure Multicast. In J. S. Vitter, P. G. Spirakis, and M. Yannakakis, editors, *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 580–589. ACM, 2001.
- [24] O. Goldreich, S. Micali, and A. Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In A. V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229. ACM, 1987.
- [25] M. Hirt, J. B. Nielsen, and B. Przydatek. Cryptographic Asynchronous Multi-party Computation with Optimal Resilience (Extended Abstract). In R. Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 322–340. Springer, 2005.
- [26] M. Hirt, J. B. Nielsen, and B. Przydatek. Asynchronous Multi-Party Computation with Quadratic Communication. In *ICALP*, volume 5126 of *Lecture Notes in Computer Science*, pages 473–485. Springer, 2008.
- [27] M. Hirt and P. Raykov. Multi-valued Byzantine Broadcast: The  $t < n$  Case. In *ASIACRYPT*, volume 8874 of *Lecture Notes in Computer Science*, pages 448–465. Springer, 2014.
- [28] A. Jaffe, T. Moscibroda, and S. Sen. On the Price of Equivocation in Byzantine Agreement. In *PODC*, pages 309–318. ACM, 2012.
- [29] R. Kapitza, J. Behl, C. Cachin, T. Distler, S. Kuhnle, S. V. Mohammadi, W. Schröder-Preikschat, and K. Stengel. CheapBFT: Resource-efficient Byzantine Fault Tolerance. In *EuroSys*, pages 295–308. ACM, 2012.
- [30] J. Katz, C. Y. Koo, and R. Kumaresan. Improving the Round Complexity of VSS in Point-to-point Networks. *Inf. Comput.*, 207(8):889–899, 2009.
- [31] D. Levin, J. R. Douceur, J. R. Lorch, and T. Moscibroda. TrInc: Small Trusted Hardware for Large Distributed Systems. In *NSDI*, pages 1–14. USENIX Association, 2009.
- [32] G. Liang and N. H. Vaidya. Error-free Multi-valued Consensus with Byzantine Failures. In *PODC*, pages 11–20. ACM, 2011.
- [33] A. Patra and C. Pandu Rangan. Communication Optimal Multi-valued Asynchronous Broadcast Protocol. In *LATINCRYPT*, volume 6212 of *Lecture Notes in Computer Science*, pages 162–177. Springer, 2010.
- [34] Arpita Patra. Error-free multi-valued broadcast and byzantine agreement with optimal communication complexity. In *Principles of Distributed Systems - 15th International Conference, OPODIS 2011, Toulouse, France, December 13-16, 2011. Proceedings*, pages 34–49, 2011.
- [35] M. C. Pease, R. E. Shostak, and L. Lamport. Reaching Agreement in the Presence of Faults. *J. ACM*, 27(2):228–234, 1980.
- [36] B. Pfitzmann and M. Waidner. Information-theoretic Pseudosignatures and Byzantine Agreement for  $t \geq n/3$ . Technical Report RZ 2882 (#90830), IBM Research, 1996.
- [37] T. Rabin and M. Ben-Or. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority (Extended Abstract). In D. S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*, pages 73–85. ACM, 1989.
- [38] A. Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, 1979.
- [39] S. Toueg. Randomized Byzantine Agreements. In *PODC*, pages 163–178. ACM, 1984.
- [40] R. Turpin and B. A. Coan. Extending Binary Byzantine Agreement to Multivalued Byzantine Agreement. *Inf. Process. Lett.*, 18(2):73–76, 1984.

# Probabilistic Byzantine Tolerance Scheduling in Hybrid Cloud Environments

Luciana Arantes  
 Sorbonne Universités, UPMC,  
 CNRS, Inria, LIP6  
 4 pl. Jussieu 75005 Paris,  
 France  
 Luciana.Arantes@lip6.fr

Olivier Marin  
 Computer Science, NYU  
 1555 Century Av., Pudong,  
 Shanghai, China  
 ogm2@nyu.edu

Roy Friedman  
 Computer Science, Technion  
 Haifa 32000, Israel  
 roy@cs.technion.ac.il

Pierre Sens  
 Sorbonne Universités, UPMC,  
 CNRS, Inria, LIP6  
 4 pl. Jussieu 75005 Paris,  
 France  
 Pierre.Sens@lip6.fr

## ABSTRACT

This work explores scheduling challenges in providing probabilistic Byzantine fault tolerance in a hybrid cloud environment, consisting of nodes with varying reliability levels, compute power, and monetary cost. In this context, the probabilistic Byzantine fault tolerance guarantee refers to the confidence level that the result of a given computation is correct despite potential Byzantine failures. We formally define a family of such scheduling problems distinguished by whether they insist on meeting a given latency limit and trying to optimize the monetary budget or vice versa. For the case where the latency bound is a restriction and the budget should be optimized, we present several heuristic protocols and compare between them using extensive simulations.

## Keywords

Byzantine, scheduling, hybrid cloud environments

## 1. INTRODUCTION

High performance distributed computing (HPDC) is typically obtained by breaking large computational problems offering trivial parallelism into multiple independent compute tasks and scheduling each task to be executed in a distributed environment. This enables solving heavy computational tasks using commodity hardware and operating systems.

With cloud technology, it is common to submit computations to virtual machines hosted in the cloud. The benefit of clouds includes their increased dependability and availability. However, cloud usage for heavy computations involves

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

*ICDCN '17, January 04-07, 2017, Hyderabad, India*

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007770>

substantial financial costs.

This motivates exploring hybrid computing architectures that combine desktop Grids with cloud hosted computing. In such a system, a large fraction of the computation is performed by donated machines, which significantly reduces the cost to the owner of the computation. Yet, when donated machines fail to ensure timely reliable completion, parts of the computation can be transferred to the cloud where they are ensured to obtain enough resources to complete.

Unfortunately, donated computers suffer from a non-negligible probability that some of them will not always return correct answers, i.e., act in a Byzantine manner. Such behavior might result, e.g., from malice on behalf of the owner of the machine, from intrusions to the machine, or from faults and bugs in hardware and software.

On the other hand, cloud servers and their VMs are likely to be more dependable than regular home machines. This is because cloud providers have a monetary incentive to protect their infrastructure from intrusions and to maintain their hardware and operating systems up to date. However, these cannot completely rule out intrusions and other forms of Byzantine hardware.

Finally, with recent trusted computing hardware, cloud providers can maintain a smaller set of fully trusted machines. Given the higher cost of trusted computing hardware, it is sensible to assume that these nodes will be scarce and their usage will be considerably more expensive than using standard cloud nodes.

*Our contributions.* First, we define a probabilistic hybrid computing model for HPDC composed of both home donated machines and cloud nodes. In this model, each computational task has a minimal required reliability level, a latency bound, and a budget. Similarly, each node has a known computing speed, monetary cost, and reliability reputation. We distinguish between home donated nodes which are very cheap, but are also not very reliable, standard cloud nodes which are much more dependable and powerful, but are more expensive, and cloud fully trusted nodes, which are completely dependable, but much more costly than the others and are slower than the standard cloud nodes.

Second, we define corresponding scheduling optimization problems that need to ensure that computational tasks meet their requirements. In all of them, the scheduler's goal is to find compute nodes that can return a reply whose correctness is above a given reliability threshold. However, they vary in the latency and budget guarantees they provide. One set of problems ensure bounded latency and attempt to minimize the required budget while the others ensure a bounded budget and try to minimize the latency.

Third, we devise several heuristic protocols for these problems. For lack of space, this paper focuses on solving the variant in which the latency bound must be observed while trying to minimize the required budget.

Last, we evaluate the performance of our protocols by simulation. Results show that in all tested workloads, an adaptive protocol, which schedules tasks on different groups of nodes, is efficient and successfully allocates all tasks within the latency constraints.

The rest of this paper is organized as follows: We survey related work in Section 2. Model assumptions are defined in Section 3. The problem statements as variations of optimization problems are presented in Section 4. Section 5 describes our scheduling protocols for bounded latency. Section 6 presents performance evaluation results of these protocols through extensive simulations. Finally, Section 7 concludes the paper.

## 2. RELATED WORK

There is vast literature on *Byzantine fault tolerance* (BFT), mainly w.r.t. *consensus* and *state replication*, e.g., [8, 12, 13, 15, 17] to name a few. In particular, multiple works have studied the notion of probabilistic consensus, where either safety is always ensured and termination becomes probabilistic or vice versa, e.g., [5–7, 10, 16, 21, 24, 25, 30]. Such level of BFT requires a minimum of  $3f + 1$  replicas, resulting in significant resource overheads, especially when  $f$  can be larger than 1.

Reducing this inherent overhead has been explored in several ways. One example is the introduction of wormholes and other types of trusted hardware components [31]. Another idea is separating ordering from execution as proposed in [32], such that the data itself is replicated only on  $2f + 1$  nodes. Yet, this requires a separate ordering service, often implemented by traditional BFT protocols, so the savings depends on data being much larger than control and metadata sizes.

Several works have explored how to harden high performance distributed computing environments, such as [2, 3], against Byzantine failures [1, 9, 11, 28, 29]. Our model differs from theirs in the following ways: At the node level, we incorporate in our model the reliability level of each node, the computational power of each node, and the monetary cost of using each node. In particular, we assume a hybrid execution model composed of nodes from several levels of reliability, compute power, and cost. Further, at the computational task level, we combine a minimal reliability level, latency constraints, and budget constraints. Hence, the scheduling task in our work is much more involved.

A pull-based scheduler for hybrid distributed computing infrastructure (Desktop, Grid, and Cloud nodes) that relies on multi-criteria decision-making method for task assignment is presented in [23]. The multi-criteria method computes for each task a set of criteria according to the

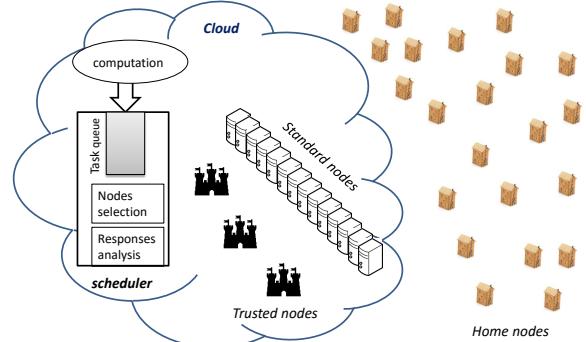


Figure 1: Hybrid cloud architecture

characteristics of the node requesting a task. Among the criteria, the authors propose the expected time and cost to complete the task, as well as the estimated error impact of scheduling the task to the pulling node, taking into account both its reputation and the size of the task. Similarly to us, their aim is to find an optimal scheduling strategy in a hybrid environment that satisfies user constraints expressed in terms of cost, price, and reliability. To this end, they apply a filtering methodology denoted SOFT. Yet, contrarily to our approach, their scheduler is pull-based where idle nodes ask for task assignment, there is no Byzantine behavior, and the scheduler does not apply a per group selection approach as our adaptive protocol does in order to avoid combinatory explosion.

Some hybrid Cloud platforms combine public and private Clouds. Some works [19, 20] focus on resource provisioning in the presence of failures. Private cloud nodes are usually considered free for users but prone to failures whereas public cloud nodes are trusted (failure-free) but users must pay to use them. As in our approach, task scheduling decisions depend on reliability requirements and cost constraints. For instance, in [20], the authors consider that failures in private cloud can be correlated in space and time. Hence, jobs that request more than a number of VM threshold or last more than a deadline threshold are redirected to public cloud nodes.

## 3. SYSTEM AND THREAT MODELS

We consider a hybrid cloud architecture in which *computing tasks* continuously arrive and need to be scheduled on a large pool of available *compute nodes* (physical or VMs), similar to the one depicted in Figure 1. The compute nodes include a combination of *home donated desk-tops* (a.k.a *home nodes*) as well as *cloud hosted VMs* (a.k.a *cloud nodes*). The cloud nodes are also divided into a plurality of *standard cloud nodes* and a smaller set of *fully trusted cloud nodes*. The difference between these sets of nodes is relates to their compute power, expected reliability, and availability.

That is, home nodes are expected to have relatively lower compute power and high *churn rate*, meaning that their availability and reliability are low. In particular, the probability of Byzantine behavior on their part is higher than all other nodes. Usage of home nodes is assumed to be very cheap, but not completely free of charge, as typically cloud providers charge a small amount of money for I/O outside the cloud.

Standard cloud nodes have relatively high compute power

$p$	A compute node	$B(\Gamma_\tau)$	Budget of schedule $\Gamma_\tau$
$p_i, p_j$	Distinguishing between compute nodes	$\Delta_\tau$	A latency threshold for $\tau$
$q$	A probability	$\Phi_\tau$	A budget threshold for $\tau$
$q_p$	Probability that $p$ will return an incorrect answer	$G_\tau$	Group of fully trusted nodes
$r_p$	Reliability of $p$ ( $1 - q_p$ )	$G_C$	Group of standard cluster nodes
$c_p$	Compute speed of node $p$	$G_H$	Group of home nodes
$d_p$	Cost of node $p$	$G_i$	A group of nodes of the same type
$\tau$	A compute task	$\sigma$	A run
$T_\tau$	Normalized compute time for $\tau$	$L_\sigma$	The latency of $\sigma$
$\rho_\tau$	A reliability threshold for $\tau$	$B_\sigma$	The latency of $\sigma$
$\Gamma_\tau$	A schedule for $\tau$	$S_\sigma$	Set of nodes producing replies in $\sigma$
$V(\Gamma_\tau)$	A reply values sequence for $\Gamma_\tau$	$t_i$	Time
$L(\Gamma_\tau)$	Latency of schedule $\Gamma_\tau$	$v_i$	Reply value

Table 1: Main symbols used in this paper

and can be allocated on demand. They are much more reliable than home nodes, but they might still occasionally fail including in a Byzantine manner. Utilizing standard cloud nodes involves paying a small fee.

Finally, fully trusted cloud nodes have a very low probability of suffering a *crash* failure, and never suffer from Byzantine failures. The number of fully trusted cloud nodes is bounded and their usage cost is significantly higher than using standard cloud nodes. The existence of such fully trusted nodes is backed by recent advancements in trusted cloud computing hardware [31], or alternatively can be realized by clustering standard cloud nodes using BFT techniques [8, 12, 13].

Further, the scheduler, which accepts the computing tasks and distributes them to various compute nodes, runs on a fully trusted cloud node. That is, the scheduler is assumed to be fault-tolerant, always available, and it always obeys its prescribed protocol.

The communication in the system is performed by sending and receiving messages over a communication network. The network is assumed to be authenticated and reliable, with a bounded communication latency. That is, a message is only received if it was indeed sent by some node, and the receiver always knows who the true sender of a message is.

As mentioned before, the home nodes and standard cloud nodes may occasionally act in a *Byzantine* manner. That is, while executing a compute task, each such node  $p$  may return an incorrect answer (or not answer at all) with probability  $q_p$ . We refer to the probability  $r_p = 1 - q_p$  that  $p$  returns a correct answer as the *reliability* or *reputation* of  $p$ . Notice that  $r_p$  may change overtime. When the system starts, for any home node  $p_i$  and arbitrary standard cloud node  $p_j$ ,  $r_{p_i} < r_{p_j}$ . Further, even when the reliability of nodes changes over time, the above inequality would remain true for the vast majority of home nodes and cloud nodes. Obviously, for a fully trusted node  $p$ ,  $r_p$  is always 1.

Whenever a scheduler node receives a compute task, it sends it either to a fully trusted cloud node or to multiple compute nodes. In the former case, the scheduler knows that it will get a correct answer. However, since fully trusted nodes are scarce and expensive, the scheduler often prefers the latter option. In these cases, when the replies arrive, the scheduler compares them. If they all agree, then the scheduler knows that this is the correct answer with a certainty that depends on the reputations of the chosen nodes. Otherwise, if some replies do not return within the deadline, the scheduler knows that these nodes are faulty and sends the

same compute task to additional nodes. Yet, if the replies do not match, then the scheduler knows that at least some of the nodes acted in a Byzantine manner and may send the compute task to additional nodes until it has enough probabilistic confidence in one of the replies.

Each compute task  $\tau$  has a normalized compute time  $T_\tau$  and that each compute node  $p$  has a known computing speed  $c_p$ . So when there are no failures, a task  $\tau$  that is scheduled to be computed on a node  $p$  completes its execution on  $p$  within time  $T_\tau/c_p$ . Further, each compute node  $p$  charges  $d_p$  units of money per second of computing (for home nodes  $d_p = 0$ ). Hence, the cost of computing task  $\tau$  on node  $p$  is  $\frac{T_\tau \cdot d_p}{c_p}$ .

The number of nodes needed to execute each compute task to obtain a trusted reply as well as the expected compute time and cost are the main topic of this paper.

### 3.1 A Generic Model of Hybrid Computation

In the most generic case, compute nodes are divided into multiple groups  $\{G_i\}$ , each characterized by an initial reputation value  $R_i$ , its own scheme for managing the reputation by the scheduler  $F_i$ , a range of compute power  $[C_i^{\min}, \dots, C_i^{\max}]$ , and a range of costs  $[D_i^{\min}, \dots, D_i^{\max}]$ . For each group  $G_i$  and for each node  $p \in G_i$ , the initial reputation of  $p$ ,  $r_p = R_i$ . The compute power  $c_p \in [C_i^{\min}, \dots, C_i^{\max}]$  and the computing cost  $d_p \in [D_i^{\min}, \dots, D_i^{\max}]$ . The reputation management scheme  $F_i$  is a function that is invoked by the scheduler each time a computation that  $p$  was involved in terminates and sets a new value for  $r_p$  based on its old value and all returned results for the compute task.

In the case of home machines, standard cloud nodes, and fully trusted cloud nodes, each of the above forms a group. For the fully trusted nodes  $R_i = 1$  and  $F_i(*) = 1$ . Further,  $D_i^{\min} > D_k^{\max}$  for any other  $k$ , i.e., they are the most expensive nodes. However,  $C_i^{\max} < C_k^{\min}$  when  $k$  is standard cloud nodes, as trusted hardware is assumed to be slower than commodity one.

At the other end, in the desktop group,  $R_i < R_k$  for any other group  $k$ , i.e., these nodes are the least trusted. Also,  $D_i^{\min} \leq D_i^{\max} \ll D_k$  for any other  $k$ , i.e., using these nodes is very cheap compared to the others.

Several works explored the reputation management schemes ( $F_i$ s) and its effectiveness, e.g., [4, 18, 26, 28]. For lack of space, we defer exploring the impact of  $F_i$  to future work. In particular, the simulations in Section 6 assume that each node has a fixed reliability level.

## 4. PROBLEM STATEMENT

We present two variations of an optimization problem. Specifically, the goal of the scheduler is to submit each task to one or more compute nodes such that the reliability level of the result is equal or greater to a given threshold  $\rho$ . In one variant of the problem, each application, composed by a set of tasks, gets a maximum budget allocation, and the scheduler needs to minimize the expected latency until obtaining a correct answer. In the other variant, the application has a maximal latency and the scheduler needs to minimize the cost of the computation. Yet, in order to rigorously state the problem definitions, we must first develop adequate terminology. In the definitions below, for simplicity of presentation, we ignore the transmission times of messages.

### 4.1 Framework and Terminology

### 4.1.1 Schedules

Given a compute task  $\tau$ , a *schedule*  $\Gamma_\tau$  is a sequence of tuples  $\{< t_i, p_i >\}$  representing a time  $t_i$  and a compute node  $p_i$  such that for each two tuples  $< t_i, p_i >$  and  $< t_j, p_j >$  in  $\Gamma_\tau$ , if  $p_i \neq p_j$  and  $< t_i, p_i >$  appears before  $< t_j, p_j >$  in  $\Gamma_\tau$  then  $t_i \leq t_j$ . Intuitively, the schedule indicates the starting time of the task on each compute node.

Since we assumed that each task  $\tau$  has a normalized compute time  $T_\tau$  and each node  $p$  has a compute power  $c_p$ , each tuple in  $\Gamma_\tau$  is implicitly associated with a time  $t'_i = t_i + \frac{T_\tau}{c_p}$  such that if  $p$  is correct, the scheduler is guaranteed to obtain a reply from  $p$  by  $t'_i$ . In fact, if no reply is received by  $t'_i$ , then  $p$  is assumed to be faulty.

### 4.1.2 Reply Sequences

Suppose  $\tau$  is sent to various compute nodes according to  $\Gamma_\tau$ . The replied values can be represented by a sequence  $V_{\Gamma_\tau}$  of tuples  $< t'_i, v_i >$  where  $v_i$  is the corresponding reply value arriving at time  $t'_i$ . If a reply does not arrive in time, the matching  $v_i$  is set to  $\perp$ . In case  $p_i$  is correct,  $v_i$  is the correct reply. A reply value sequence  $V_{\Gamma_\tau}$  is called *complete* if for each tuple in  $\Gamma_\tau$  there is a corresponding tuple in  $V_{\Gamma_\tau}$  and is said to be *partial* if it is a prefix of a complete reply sequence.

### 4.1.3 Runs

Next, we define a boolean *stopping function*  $ST(V_{\Gamma_\tau}) = [\text{true}, \text{false}]$ . Intuitively, this function enables the scheduler to stop the schedule prematurely, before contacting all nodes indicated by the schedule, whenever the replies obtained so far meet the condition indicated by the stopping function. To that end, we define a *stoppable schedule* to be the combination of a schedule and a corresponding stopping function. Hereafter, we only deal with stoppable schedules. Hence, whenever we write schedule, we in fact mean stoppable schedule.

A (partial) reply sequence  $V_{\Gamma_\tau}$  is called *minimal* if  $ST(V_{\Gamma_\tau}) = \text{true}$  and for each prefix  $V'$  of  $V_{\Gamma_\tau}$ ,  $ST(V') = \text{false}$ . Each combination of a schedule and a minimal reply sequence defines a *run*  $\sigma$ .

### 4.1.4 Latencies and Budgets

Let  $< t_1, * >$  be the first tuple in a schedule of a run  $\sigma$  and let  $< t_k, * >$  be the last tuple in the matching reply sequence of  $\sigma$ . We define the *latency* of  $\sigma$  (denoted  $L_\sigma$ ) to be  $t_k - t_1$ . Similarly, we define the *budget spent during*  $\sigma$  (denoted  $B_\sigma$ ) as the total cost of all compute nodes whose replies appear in the corresponding reply sequence  $V_{\Gamma_\tau}$ . That is, let  $S_\sigma$  be this set of compute nodes. We then have  $B_\sigma = \sum_{p \in S_\sigma} \frac{T_\tau d_p}{c_p}$ .

## 4.2 Two Families of Problems

We define two families of dual optimizations problems. In the first, latency must be kept below a given bound while the budget should be minimized. In the second, the budget must be kept below a given bound while latency should be minimized.

### 4.2.1 Bounded Latency

Given a task  $\tau$ , a latency threshold  $\Delta_\tau$  and reliability threshold  $\rho_\tau$  for task  $\tau$ , the scheduler's goal is to find a schedule  $\Gamma_\tau$  minimizing the corresponding budget  $B(\Gamma_\tau)$  for computing  $\tau$  while obtaining an answer whose reliability is

above  $\rho_\tau$  and the latency  $L(\Gamma_\tau)$  is at most  $\Delta_\tau$  time (assuming feasible).

### 4.2.2 Bounded Budget

Given a task  $\tau$ , a budget threshold  $\Phi_\tau$  and reliability threshold  $\rho_\tau$  for task  $\tau$ , the scheduler's goal is to find a schedule  $\Gamma_\tau$  minimizing the corresponding latency  $L(\Gamma_\tau)$  for computing  $\tau$  while obtaining an answer whose reliability is above  $\rho_\tau$  and the budget  $B(\Gamma_\tau)$  is at most  $\Phi_\tau$  (assuming feasible).

As mentioned before, due to lack of space, in this paper we focus on solving the bounded latency problem.

## 5. GREEDY BOUNDED LATENCY

In this section, we present several variants of scheduling protocols addressing the bounded latency problem. The first set of protocols always attempt to find a schedule from the same group of nodes (home, cloud, or trusted) in an iterative manner. Conversely, the other protocol employs an adaptive mechanism in which it considers all groups and chooses the cheapest option that still ensures timely termination.

### 5.1 Single Group Protocols

We identify three protocols in the single group category, nicknamed *scrooge*, *moderate*, and *cautious*. The only difference between these protocols is in the group of processes from which each of these protocols looks for its candidate nodes; scrooge only utilizes home nodes, moderate only accesses standard cloud nodes, while cautious only fully trusted nodes.

In all three protocols, the scheduler invokes the following iterative loop, as outlined in Algorithms 1, 2, 3, and 4: First, it looks for the cheapest set of available nodes from the corresponding group (home, standard cloud, or trusted) such that the probability that all of them are Byzantine is below the required reliability threshold and the maximal latency of any of them for the given compute task is below its latency requirement. The task is then sent to all chosen nodes to be computed and the scheduler waits until it receives either a reply from all of them or a timeout equal to the longest expected latency has passed. If all replies have arrived and all have the same value, then the scheduler returns this value.

Otherwise, suppose some value  $v$  has appeared in the maximal number of replies (breaking symmetry arbitrarily). The scheduler looks for another set of processes such that if they all return  $v$ , then the probability that  $v$  is the correct value is above the required threshold. Here again, the cheapest possible set that meets the remaining latency deadline is chosen. This process repeats until either the probability that some returned value is correct meets the reliability threshold, or it is not possible to find additional nodes that can compute the task within the required deadline.

To understand the calculation for *ExtraNeeded* in Algorithm 1, let's denote *ProbCorrectV* by  $A$ , *AllByzantine* by  $C$ , and the calculation  $\prod_{p_i \in S_1} (1 - r_i)$  by  $D$ . Obviously, we can write  $A = (1 - D)C$ . Similarly, we can write  $q = (1 - DX)C$  where  $X$  is the probability that all nodes in the added set (that is required) are Byzantine. In other words,  $1 - DX$  is the probability that the combined set of existing nodes that support  $v$  and all added nodes will include at least one correct node. Using simple algebra, we get that  $X = \frac{C - q}{C - A}$ , or as written in Figure 1, *ExtraNeeded* =

---

**Algorithm 1** Probabilistic Functions

---

```

1: // Returns TRUE if the probability that all nodes in  $S$  are Byzantine is  $\leq q$ 
2: function ALLBYZ( $S, q$ )
3:   return  $\prod_{p_i \in S} (1 - r_i) \leq q$ 
4:
5: // Returns the probability that at least one of the nodes in  $values$  whose value is  $v$  is correct and all nodes proposing other values are Byzantine
6: // Also, return the needed probability from additional set of supporters of  $v$  to ensure that  $v$  is correct if not already obtained
7: function PROBCORRECTVALUE( $v, values, q$ )
8:   if  $v == \perp$  then
9:     return  $(0, 0)$ 
10:     $S_1 = \{p_i | (v_i, p_i) \in values \wedge v_i == v\}$ 
11:     $AtLeastOneCorrect = 1 - \prod_{p_i \in S_1} (1 - r_i)$ 
12:     $S_2 = \{p_i | (v_i, p_i) \in values \wedge v_i \neq v\}$ 
13:     $AllByzantine = \prod_{p_i \in S_2} (1 - r_i)$ 
14:     $ProbCorrectV = AtLeastOneCorrect \cdot AllByzantine$ 
15:    if  $ProbCorrectV < q$  then
16:       $ExtraNeeded = \frac{AllByzantine - q}{AllByzantine - ProbCorrectV}$  // See explanation in text
17:      return  $(ProbCorrectV, ExtraNeeded)$ 
18:    else
19:      return  $(ProbCorrectV, 0)$ 

```

---

$$\frac{AllByzantine - q}{AllByzantine - ProbCorrectV}.$$

## 5.2 The Adaptive Protocol

As before, in the adaptive algorithm too we allow multiple sequential invocations of the task until some returned value obtains enough reliability to be considered the correct reply value. However, here we allow to switch between the different groups of nodes (home, cloud, trusted). That is, home nodes are the cheapest, but may not be able to provide a reliable answer within the deadline due to their low reliability. Trusted nodes always return a correct answer, but are expensive and scarce, and hence are likely to run out if we insist on only using them. Cloud nodes are more reliable than home nodes, but might still fail and are considerably more expensive than the home nodes.

Hence, we start with the cheapest set among all groups that is likely to produce a correct result. But, this time, we reserve enough latency so that in the worst case, we can resort to a trusted node, or to at least a collection of cloud nodes, in order to ensure reliable termination.

Obviously, trying all possible combinations of nodes to determine the optimal one is too expensive. Hence, in each iterative step we always select sets of nodes from the same  $G_i$  group. Moreover, we identify a few subsets in each group and try combinations of these subsets such that the total latency is below the threshold while the ensured reliability is above the threshold. We then choose the set whose cost is minimal among them.

Specifically, in each iteration we first identify the cheapest and fastest available trusted nodes that can compute  $T_\tau$  within the remaining deadline  $\Delta'$ . Obviously, in the first iteration, the above is done w.r.t. the entire deadline, i.e.,  $\Delta' = \Delta$ . Denote the faster of these nodes  $p_T^F$  and the cheaper  $p_T^E$  (so  $F$  stands for fast and  $E$  for economic and  $T$  for trusted). Similarly, denote the latency and budget that would be spent when executing on  $p_T^F$  by  $L_T^F$  and  $B_T^F$  respectively. In symmetry, denote the latency and budget that would be spent when executing on  $p_T^E$  by  $L_T^E$  and  $B_T^E$  respectively.

Considering these two latencies, we now identify sets of

---

**Algorithm 2** Helper Functions

---

```

1: // Add the corresponding budget and latency to a schedule
2: function FLESHOUT( $S, T$ )
3:    $budget = \sum_{p_i \in S} \frac{T_{d_i}}{c_i}$ 
4:    $latency = \max_{p_i \in S} \frac{T}{c_i}$ 
5:   return  $(S, budget, latency)$ 
6:
7: // Find a set that can compute  $T$  fast enough whose members satisfy the required reliability
8: function FINDSET( $G, T, \Delta, q$ )
9:    $cands = \{p_i \in G | T/c_i \leq \Delta\}$  // find all nodes that are fast enough
10:  if  $cands == \emptyset$  then
11:    raise no_schedule_found
12:  else
13:     $sets = \{S_i \subseteq cands | ALLBYZ(S_i, q)\}$  // All possible sets
14:     $finalists = \{S_i \in sets | \sum_{p_i \in S_i} \frac{T_{d_i}}{c_i}$  is minimal in  $sets\}$  // Filter for the cheapest
15:    if  $finalists == \emptyset$  then
16:      raise no_schedule_found
17:    else
18:      return FLESHOUT(first( $finalists, T$ )) // Return one of the cheapest
19:
20: // Send  $T$  to all members of  $set$  and wait for replies
21: function SCHEDULESTEP( $set, T, latency$ )
22:   foreach  $p_i \in set$  SEND( $p_i, T$ )
23:   wait for replies from each  $p_i \in set$  or timeout after  $latency$  time
24:    $replies =$  set of tuples  $(v_i, p_i)$  of replied values and corresponding nodes who returned these values
25:   for nodes  $p_i$  that failed to return any value by the  $latency$  timeout, the value is  $v_i = \perp$ 
26:   return  $replies$ 

```

---

standard Cloud nodes that can potentially compute  $T_\tau$  with the required reliability (if all return the same value) within  $\Delta'_F = \Delta' - L_T^F$  and  $\Delta'_E = \Delta' - L_T^E$ , respectively. For each of these corrected latency bounds, we search for the cheapest and fastest sets of cloud nodes. Denote these sets of cloud nodes  $S_C^{FF}, S_C^{FE}, S_C^{EF}$ , and  $S_C^{EE}$ . Similarly, we denote the corresponding latency and budget to be spent by each of these sets by  $L_C^{FF}, B_C^{FF}, L_C^{FE}, B_C^{FE}, L_C^{EF}, B_C^{EF}, L_C^{EE}$  and  $B_C^{EE}$ .

Finally, for each of the above latencies, we identify sets of home nodes that can potentially compute  $T_\tau$  with the required reliability level (if all return the same value) within  $\Delta'_{FF} = \Delta'_F - L_C^{FF}$ ,  $\Delta'_{FE} = \Delta'_F - L_C^{FE}$ ,  $\Delta'_{EF} = \Delta'_E - L_C^{EF}$ , and  $\Delta'_{EE} = \Delta'_E - L_C^{EE}$ , respectively. Here, it is enough to identify the cheapest such sets of home nodes, denoted  $S_H^{FF}, S_H^{FE}, S_H^{EF}$ , and  $S_H^{EE}$ . Their corresponding latency and budget are denoted  $L_H^{FF}, B_H^{FF}, L_H^{FE}, B_H^{FE}, L_H^{EF}, B_H^{EF}, L_H^{EE}$  and  $B_H^{EE}$ .

Considering  $p_T^F = S_T^F$  and  $p_T^E = S_T^E$ , once we have all the 10 sets, the scheduler picks the one whose budget is smallest, sends the computation to that set and waits either for all replies or a timeout. The scheduler then checks if some value has enough support to be deemed correct with the required reliability threshold. If yes, then this value is returned. Otherwise, the scheduler continues to the next iteration. The pseudo-code for this protocol appears in Algorithms 5 and 6.

## 6. EVALUATION

We evaluate our algorithms and show the advantage of the adaptive approach. Our experiments consist of simulating the various protocols described in Section 5 along the metrics defined in Section 6.1 below. The parameters for the

---

**Algorithm 3** Parameterized Single Group

---

```

1: // Schedule  $T$  whose deadline is  $\Delta$  on nodes from  $G$  with reliability  $\geq \rho$ 
2: function SCHEDULE( $G, T, \Delta, \rho$ )
3:    $values = \emptyset$ 
4:    $reqrel = 1 - \rho$ 
5:   loop
6:     try ( $set, budget, latency$ ) = FINDSET( $G, T, \Delta, reqrel$ )
7:     catch no_schedule_found raise no_schedule_found
8:      $values = values \cup$  SCHEDULESTEP( $set, T, latency$ )
9:     if  $|values| == 1 \wedge$  this value  $\neq \perp$  then
10:        $\triangleright$  If all nodes replied the same value, return it
11:       return first(first( $values$ ))
12:     else
13:        $\triangleright$  If the probability that the majority value  $v$  is correct is high enough, return it. Otherwise, try collecting additional answers from the cheapest set of additional nodes from  $G$  such that they can return the reply in the remaining time before the deadline and if they all return  $v$  as well, it will be certain that  $v$  is correct
14:        $v =$  the most frequent value in  $values$ 
15:        $\Delta = \Delta - latency$ 
16:       ( $rel, reqrel$ ) = PROBCORRECTVALUE( $v, values, \rho$ )
17:       if  $rel \geq \rho$  then
18:         return  $v$ 

```

---

**Algorithm 4** The Specific Single Group Instances

---

```

1: function SCROOGE( $T_\tau, \Delta_\tau, \rho$ )
2:   return SCHEDULE( $G_H, T_\tau, \Delta_\tau, \rho$ )
3:
4: function MODERATE( $T_\tau, \Delta_\tau, \rho$ )
5:   return SCHEDULE( $G_C, T_\tau, \Delta_\tau, \rho$ )
6:
7: function CAUTIOUS( $T_\tau, \Delta_\tau, \rho$ )
8:   return SCHEDULE( $G_T, T_\tau, \Delta_\tau, \rho$ )

```

---

simulations are detailed in Section 6.2.

## 6.1 Metrics

We define the following metrics for comparing the performance of the various protocols: (i) *Budget (financial cost)*: the total budget actually spent by the protocol; (ii) *Completion time (response time)*: The overall completion time of the application; (iii) *Computation time*: The difference between the time a task is submitted and the time it responds (both the average and the distribution); (iv) *Task deadlines satisfaction (fail ratio)*: Percentage of task deadlines that were satisfied w.r.t. latency (deadline) and reliability constraints. A failure of a task is due either to wrong answers from Byzantine nodes which delay the response time or an overload of compute nodes; (v) *Distribution of scheduled nodes*: The total number of home, standard cloud, and fully trusted nodes used by each of the protocols; (vi) *Scheduler compute time*: The amount of CPU time used by the protocol to schedule all the tasks. This metric indicates the computational complexity of the protocol, which is important for scalability.

## 6.2 Simulation setup

Our simulations are conducted with Matlab<sup>1</sup>. Since the proposed algorithms need to compute all subsets of nodes that satisfy a criteria of cost and/or time which may lead to a combinatorial explosion, an exhaustive search of all subsets is too costly. For instance, assigning 1,000 tasks with normalized duration of 180 seconds to 300 nodes by the Moderate algorithm takes 2887 seconds on a 2 cores Intel core I7 at

<sup>1</sup><http://mathworks.com/>

---

**Algorithm 5** Helper Function for Adaptive Algorithm

---

```

1: // Returns the fastest and cheapest sets of nodes from group  $G$  than can compute  $T$  with reliability  $\geq \rho$  and latency  $\leq \Delta$ 
2: function FIND2SETS( $G, T, \Delta, \rho$ )
3:    $cands = \{p_i \in G | T/c_i \leq \Delta\}$   $\triangleright$  Find qualifying nodes in  $G$ 
4:   if  $cands == \emptyset$  then
5:     return  $\{(\emptyset, 0, 0), (\emptyset, 0, 0)\}$ 
6:   else
7:      $\triangleright$  Identify the cheapest set, break ties arbitrarily
8:      $sets = \{S_i \subseteq cands | BYZ(S_i, \rho)\} \wedge S_i$  is minimal such set}
9:      $cheapest = \{S_i \in sets | \sum_{p_i \in S_i} \frac{T_{d_i}}{c_i}$  is minimal in  $sets\}$ 
10:    if  $|cheapest| > 1$  then
11:       $cheapest = S_i | \min_{p_i \in S_i} c_i$  is maximal in  $cheapest$   $\triangleright$  break symmetry arbitrarily
12:    else
13:       $cheapest = FIRST(cheapest)$ 
14:       $\triangleright$  Identify the fastest set, break ties arbitrarily
15:       $fastests = \{S_i \in sets | \min_{p_i \in S_i} c_i$  is maximal in  $sets\}$ 
16:      if  $|fastests| > 1$  then
17:         $fastest = S_i | \sum_{p_i \in S_i} \frac{T_{d_i}}{c_i}$  is minimal in  $fastests$   $\triangleright$  break symmetry arbitrarily
18:      else
19:         $fastest = FIRST(fastests)$ 
20:         $\triangleright$  Return found sets with their budget and latency
21:        if  $cheapest == fastest$  then
22:          return {FLESHOUT( $cheapest, T$ ),  $(\emptyset, 0, 0)$ }
23:        else
24:          return {FLESHOUT( $cheapest, T$ ), FLESHOUT( $fastest, T$ )}

```

---

1.8GHz. Inspired by the power of two random choices [22], we circumvent this combinatorial explosion by random sampling of subsets and then choosing among them the one that satisfies the criteria. For instance, the 10 sampling version of the Moderate algorithm generates results which are 93% close to the original algorithm's in less than 0.5 second. Consequently, the performance results shown here correspond to the modified versions of our algorithms that use a 10 random sampling.

*Nodes setting.* We consider 3 sets of nodes (*Trusted*, *Cloud*, and *Home*). Each of them is associated to some computing power and financial cost per hour.

*Trusted nodes.* These secure cloud nodes do not need replicated execution. Alas, they are more expensive and slower than Cloud nodes. Each Trusted node has the processing power of a medium Home node with 8 ECUs (EC2 Computing Units). One ECU provides equivalent CPU capacity of 1.0-1.2 GHz 2007 Opteron. A Trusted node cost corresponds to a high power node in Amazon (c4.8xlarge), which is 1.763 dollars per hour according to EC2 pricing (<https://aws.amazon.com/ec2/>).

*Cloud nodes.* These cloud nodes are less secure but more powerful than Trusted nodes. In our simulations, Cloud nodes consist of three types of EC2 Amazon nodes from medium to high computing performance (c4.large, c4.xlarge, and c4.2xlarge instances) corresponding to Intel Xeon E5-2666v3 with 8, 16, and 31 ECUs respectively. The EC2 computing cost per hour for these node types is 0.11, 0.22, and 0.441 dollars respectively.

*Home nodes.* These home nodes are cheaper and less trusted than the other two kinds of nodes. Their compute power varies from 2 to 16 equivalent ECUs (2, 4, 8, and 16). To estimate their cost, we consider their electric consumption. We assume that each node consumes 100 Watt per hour and that the cost of energy is between 10 and 40 cents per KW/h (the average cost of energy in north America and

---

**Algorithm 6** Adaptive Algorithm

---

```

1: Main code:
2:  $values = \emptyset$ 
3:  $reqrel = 1 - \rho$ 
4: loop
5:    $(S_T^E, B_T^E, L_T^E, S_C^F, B_C^F, L_T^F) = \text{FIND2SETS}(G_T, T_\tau, \Delta, reqrel)$ 
6:    $(S_C^{EE}, B_C^{EE}, L_C^{EE}, S_C^{FE}, B_C^{FE}, L_C^{FE}) = \text{FIND2SETS}(G_C, T_\tau, \Delta - L_T^E, reqrel)$ 
7:    $(S_C^{EF}, B_C^{EF}, L_C^{EF}, S_C^{FF}, B_C^{FF}, L_C^{FF}) = \text{FIND2SETS}(G_C, T_\tau, \Delta - L_T^F, reqrel)$ 
8:    $(S_H^{EE}, B_H^{EE}, L_H^{EE}, \dots, \dots) = \text{FIND2SETS}(G_H, T_\tau, \Delta - L_C^{EE}, reqrel)$ 
9:    $(S_H^{EF}, B_H^{EF}, L_H^{EF}, \dots, \dots) = \text{FIND2SETS}(G_H, T_\tau, \Delta - L_C^{EF}, reqrel)$ 
10:   $(S_H^{FE}, B_H^{FE}, L_H^{FE}, \dots, \dots) = \text{FIND2SETS}(G_H, T_\tau, \Delta - L_C^{FE}, reqrel)$ 
11:   $(S_H^{FF}, B_H^{FF}, L_H^{FF}, \dots, \dots) = \text{FIND2SETS}(G_H, T_\tau, \Delta - L_C^{FF}, reqrel)$ 
12:   $set = S \in \{S_T^E, S_T^F, S_C^{EE}, S_C^{EF}, S_C^{FF}, S_H^{EE}, S_H^{EF}, S_H^{FF}\}$ 
     $S \neq \emptyset \wedge$  the corresponding budget is minimal
13:   $\triangleright L_S$  denotes the latency of  $S$ 
14:  if  $set == \emptyset$  then
15:    raise no_schedule_found
16:   $values = values \cup \text{SCHEDULESTEP}(set, T_\tau, L_S)$ 
17:  if  $|values| == 1 \wedge$  this value  $\neq \perp$  then
18:     $\triangleright$  If all nodes replied the same value, return it
19:    return first(first(values))
20:  else
21:     $\triangleright$  If the probability that the majority value  $v$  is correct is high enough,
       return it. Otherwise, try collecting additional answers from the
       cheapest set of additional nodes from  $G$  such that they can return
       the reply in the remaining time before the deadline and if they
       all return  $v$  as well, it will be certain that  $v$  is correct
22:     $v =$  the most frequent value in  $values$ 
23:     $\Delta = \Delta - L_S$ 
24:     $(rel, reqrel) = \text{PROBCORRECTVALUE}(v, values, \rho)$ 
25:    if  $rel \geq \rho$  then
26:      return v

```

---

Europe). Their resulting costs are 0.01, 0.015, 0.02, and 0.04 dollar per hour.

For our simulation experiments, we consider an environment with 5,000 Home nodes, 500 Cloud nodes, and 50 Trusted nodes. In particular, there are 10 times more Home nodes than Cloud nodes and 10 times more Cloud nodes than Trusted nodes. Further, the percentage of Byzantine nodes in the Home nodes group is higher than in the Cloud nodes group while correct nodes in the latter have higher reputation than in the former. Table 2 summarizes our nodes setting for the three groups of nodes. In the table, the reputation value of correct is denoted  $r$  and Byzantine nodes  $rb$ , while the percentage of Byzantine nodes is denoted  $b$ .

	Trusted nodes	Cloud nodes	Home nodes
Number	50	500	5000
ECU	8	8/16/31	2/4/8/16
Cost (\$/h)	1.763	0.11/0.22/0.441	0.01/0.015/0.02/0.04
$b$	0	5%	15%
$r$	N/A	0.99	0.95
$rb$	N/A	0.2	0.1

Table 2: Hybrid cloud configuration

**Application setting.** We consider a bag of tasks (BoT) application representative of private Home Grid deployments [14]. The reliability threshold is  $\rho_\tau = 0.999$ .

We generated two scenarios with different loads with re-

Workload	# tasks	mean	stddev	min	max
low	5,000	179.9923	4.1528	162.307	194.5323
high	20,000	180.0023	4.1274	162.307	196.209

Table 3: Workload configurations

Algorithm	Time (sec)	Cost (\$)	Sched. time (sec.)	fail ratio
Cautious	476.98	11.588	0.041	79 %
Moderate	477.32	11.979	0.555	0.38 %
Scrooge	454.79	3.189	2.153	0 %
Adaptive	443.77	2.884	5.683	0 %

Table 4: Time and cost for protocols with low load

gard to the number of concurrently submitted tasks: low load (5,000 tasks) and high load (20,000 tasks), as summarized in Table 3. The duration of tasks follow a normal distribution with a mean duration around 180 seconds on one ECU. Both the mean duration of tasks and the standard deviation are set to be the same as the ones of BoT application deployed on the Home Grid of the University of Notre Dame [27]. The deadline of every task is 2.5 times the completion time on one ECU.

### 6.3 Low workload evaluation

Table 4 summarizes the cost, completion time, fail ratio, and the time spent by the scheduler for the four algorithms. We observe that Scrooge has the lowest cost among the single group algorithms. Scrooge is competitive since its nodes are cheap and abundant so they can easily handle the load within the deadline constraints. Moderate and Scrooge allocate additional nodes when wrong responses are received: on average, a task is replicated 4.225 times in the case of Home nodes and 2.227 times in the case of Cloud nodes. On the other hand, even if Trusted nodes are never replicated, they cannot ensure the execution deadline of most of the tasks. The adaptive protocol also succeeds to execute all the tasks within their deadline. Compared to the cheapest single protocol, i.e., Scrooge, the response time for the application is 2.48% lower. However, even if the cost of the adaptive protocol is 10.58% smaller than Scrooge, we cannot conclude that the adaptive protocol is always the cheapest since the difference of costs is due to the heterogeneity of nodes and the load distribution.

Table 5 shows the distributions of load in terms of the number of tasks executed by nodes of different groups. A Trusted node executes 21 tasks and then fails to satisfy deadlines. The load is unbalanced in both Cloud and Home nodes since the power of nodes is highly heterogeneous. Powerful nodes execute a large number of tasks. In both Scrooge and Adaptive, Home nodes are not used since the number of tasks is relatively low.

Figure 2 gives the computation time of each task. We observe that only 1,050 tasks are completed before their

Algorithm	# nodes	mean	stddev	min	max
Cautious	50	21	0	21	21
Moderate	500	22.186	11.8977	1	41
Scrooge	4408	4.225	3.5449	0	24
Adaptive	3375	4.232	4.3195	0	31

Table 5: Distribution of tasks on nodes with low load

Algorithm	Time (sec)	Cost (\$)	Sched. time (sec.)	fail ratio
Cautious	476.98	11.588	0.115	94.75 %
Moderate	485.35	16.594	1.476	66.23 %
Scrooge	492.94	11.064	3.770	14.04 %
Adaptive	471.52	17.816	13.391	0 %

Table 6: Time and cost with high load

Algorithm	# nodes	mean	stddev	min	max
Cautious	50	21	0	21	21
Moderate	500	30.358	11.997	10	65
Scrooge	4999	14.413	8.4422	0	39
Adaptive					
Trust set	45	2.32	1.53	0	7
Cloud set	377	12.38	8.17	0	31
Home set	4949	14.56	10.29	0	40

Table 7: Distribution of tasks on nodes with high load

deadline in the Cautious algorithm, the Moderate algorithm fails to complete 19 tasks, and only Scrooge and Adaptive succeed in completing all tasks. In the adaptive approach, we observe that Trusted and Home nodes execute all the tasks, i.e., no task is assigned to nodes of the Cloud group. Interestingly, even in the relatively low load configuration, the application benefits from the adaptive algorithm since it distributes the load among the cheapest nodes but assigns a task to a Trusted node whenever the execution of this task cannot be performed on the Home group.

#### 6.4 High workload evaluation

Table 6 summarizes the results with heavy load. Since there is no sharing of tasks between the three groups for the single group approach, the three algorithms fail to complete the deadline for a large number of tasks. Having a higher number of nodes (20,000), Scrooge is clearly the most efficient with regard to fail deadline ratio. However, costs are not representative since the algorithms do not succeed in executing all the tasks. The average costs in dollars per scheduled tasks are 0.011, 0.0025, 0.00064 in Cautious, Moderate, and Scrooge respectively. The adaptive algorithm succeeds in executing all the tasks within their deadline. Compared to the single group algorithms, the response time for the application is lower than Scrooge and the global cost is higher since all tasks are scheduled. The average cost per task is 0.00089 dollars. The cost per task is 38.4% higher than with Scrooge since the adaptive scheduler needs to use expensive Trusted nodes in this scenario.

Table 7 provides the tasks distribution per node for each group. The load distributions are similar to the low load scenario. The algorithms favor choosing powerful nodes; even if the latter are more costly, they succeed more often to complete tasks within the deadline. Thus, their short response time reduces costs. Notice that the adaptive algorithm uses all three sets of nodes.

Figure 3 shows the computation time of each task. In the single group approach, a large number of tasks cannot meet their deadline and are, thus, not scheduled. As shown in Figure 3(a), after 1,000 tasks, no other task is scheduled by the Cautious protocol. The gaps without load after the number of scheduled tasks reaches 5,500 in Figure 3(b) indicate that many tasks could not be scheduled in the Moderate protocol. Some tasks, denoted “faulty tasks”, are scheduled on Cloud and Home nodes but fail to meet their deadline because

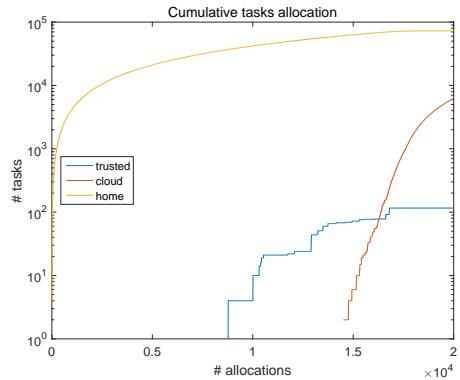


Figure 4: Task allocation using adaptive algorithm

of the high number of additional nodes needed to obtain enough correct answers. Among the 6,753 tasks it schedules, Moderate generates 173 faulty tasks (2.56%) while Scrooge generates 332 faulty tasks among its 17,253 scheduled tasks (1.92%). Figure 3(d) gives the computation time of each task with Adaptive. We observe that Trust nodes start participating in the computation after 8,000 scheduled tasks. After 15,000 tasks, Trust and Home nodes become overloaded and then tasks are mainly scheduled on the nodes of the Cloud group. Ultimately, 17,308 tasks are scheduled on Home nodes, 2,692 on Cloud nodes, 116 on Trusted nodes. Among these tasks, 109 are “hybrid”, i.e., replicated both on Home and Trusted nodes. Figure 4 gives the cumulative number of replicas associated to each set.

## 7. CONCLUSION

We have explored a hybrid computing model composed of groups of home, standard cloud, and trusted cloud nodes, where each node has a given computing speed, monetary cost, and reliability reputation. We presented four protocols (Scrooge, Moderate, Cautious, and Adaptive) for scheduling tasks in such environments aiming at ensuring BoT applications’ requirements in terms of bounded latency and reliability, while minimizing their spent budget. All protocols were evaluated by simulation under both low and high workloads.

Performance results show that in both workloads, Scrooge (resp., Cautious) is the cheapest (most expensive) and has the smallest (resp., the highest) task deadline fail ratio. Yet, it takes more (resp., less) time to schedule tasks since, due to the low (resp., high) reputation of the home (resp., trusted) nodes, a single task must be submitted to more (resp., just one) nodes. In contrast, in high workload, these metrics increase, and Scrooge can no longer ensure all tasks’ deadlines.

In both workloads, all tasks are successfully scheduled by Adaptive and meet their deadline since Adaptive distributes the load among the cheapest nodes, but assigns tasks to more reliable nodes whenever this task cannot be executed in Home nodes. Further, its response time is slightly faster than with Scrooge. Such good results confirm the advantage of the adaptive protocol.

As future work, we intend to propose and evaluate protocols for the Bounded Budget scheduling optimizing problem, described in section 4.2. We also plan to investigate the impact of different reputation management strategies on the protocols’ performance.

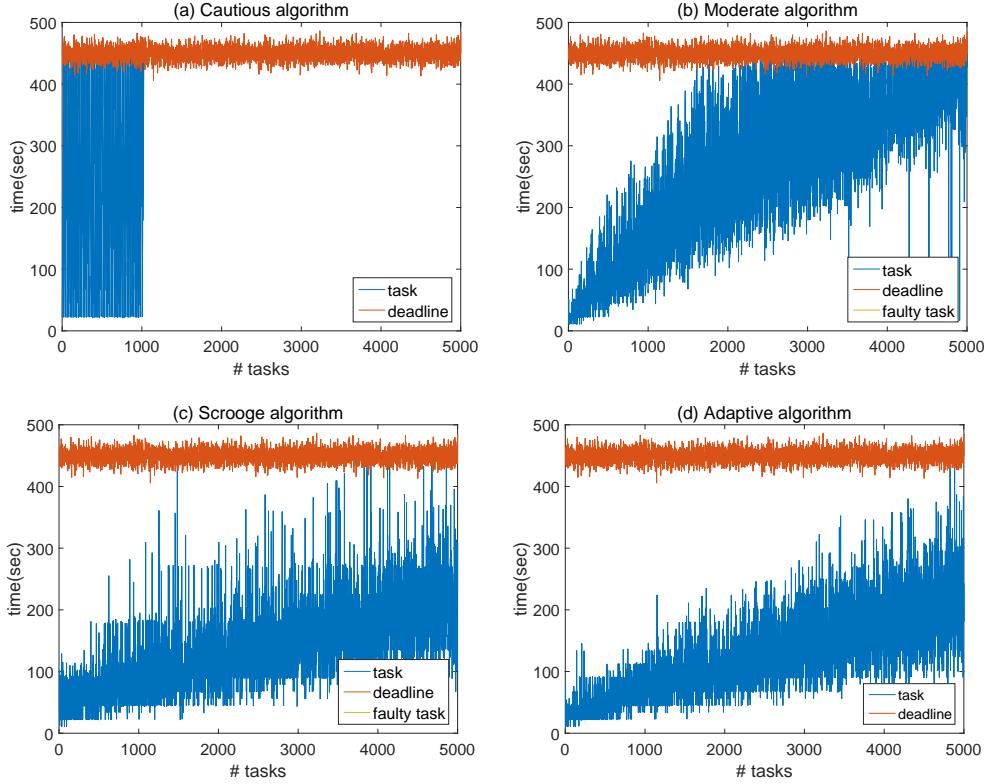


Figure 2: Computation time of tasks with low load

## 8. REFERENCES

- [1] A. Agbaria and R. Friedman. A Replication- and Checkpoint-Based Approach for Anomaly-Based Intrusion Detection and Recovery. In *IEEE Int. Conf. on Distributed Computing Systems Workshops*, pages 137–143, 2005.
- [2] D. P. Anderson. BOINC: A system for public-resource computing and storage. In *Proc. of the 5th IEEE/ACM Int. Workshop on Grid Computing*, pages 4–10, 2004.
- [3] D. P. Anderson, J. Cobb, E. Korppela, M. Lebofsky, and D. Werthimer. SETI@home: An experiment in public-resource computing. *C. ACM*, 45(11):56–61, 2002.
- [4] L. Arantes, R. Friedman, O. Marin, and P. Sens. Probabilistic byzantine tolerance for cloud computing. In *Proc. of the IEEE SRDS*, 2015.
- [5] J. Aspnes. Lower bounds for distributed coin-flipping and randomized consensus. *J. of the ACM*, 45:559–568, 1997.
- [6] H. Attiya and K. Censor. Tight bounds for asynchronous randomized consensus. *J. ACM*, 55(5):20:1–20:26, 2008.
- [7] H. Attiya and K. Censor-Hillel. Lower bounds for randomized consensus under a weak adversary. *SIAM J. Comput.*, 39(8):3885–3904, 2010.
- [8] P.-L. Aublin, R. Guerraoui, N. Knežević, V. Quémé, and M. Vukolić. The next 700 bft protocols. *ACM Trans. Comput. Syst.*, 32(4):12:1–12:45, 2015.
- [9] A. Bondavalli, S. Chiaradonna, F. Giandomenico, and J. Xu. An Adaptive Approach to Achieving Hardware and Software Fault Tolerance in a Distributed Computing Environment. *J. of Sys. Arch.*, 47(9):763–781, 2002.
- [10] G. Bracha. An asynchronous  $[(n - 1)/3]$ -resilient consensus protocol. In *Proc. of the ACM PODC*, pages 154–162, 1984.
- [11] Y. Brun, G. Edwards, J. Y. Bang, and N. Medvidovic. Smart Redundancy for Distributed Computation. In *Proc. of the 31st IEEE ICDCS*, pages 665–676, 2011.
- [12] M. Castro and B. Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.*, 20(4):398–461, 2002.
- [13] A. Clement, M. Kapritsos, S. Lee, Y. Wang, L. Alvisi, M. Dahlin, and T. Riche. Upright cluster services. In *Proc. of the ACM SIGOPS 22Nd SOSP*, pages 277–290, 2009.
- [14] S. Delamare, G. Fedak, D. Kondo, and O. Lodygensky. Spequulos: a qos service for hybrid and elastic computing infrastructures. *Cluster Computing*, 17(1):79–100, 2014.
- [15] C. Dwork, N. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, 1988.
- [16] R. Friedman, A. Mostefaoui, and M. Raynal. Simple and efficient oracle-based consensus protocols for asynchronous byzantine systems. *IEEE Trans. on Dependable and Secure Computing*, 2(1):46–56, 2005.
- [17] R. Garcia, R. Rodrigues, and N. Preguiça. Efficient

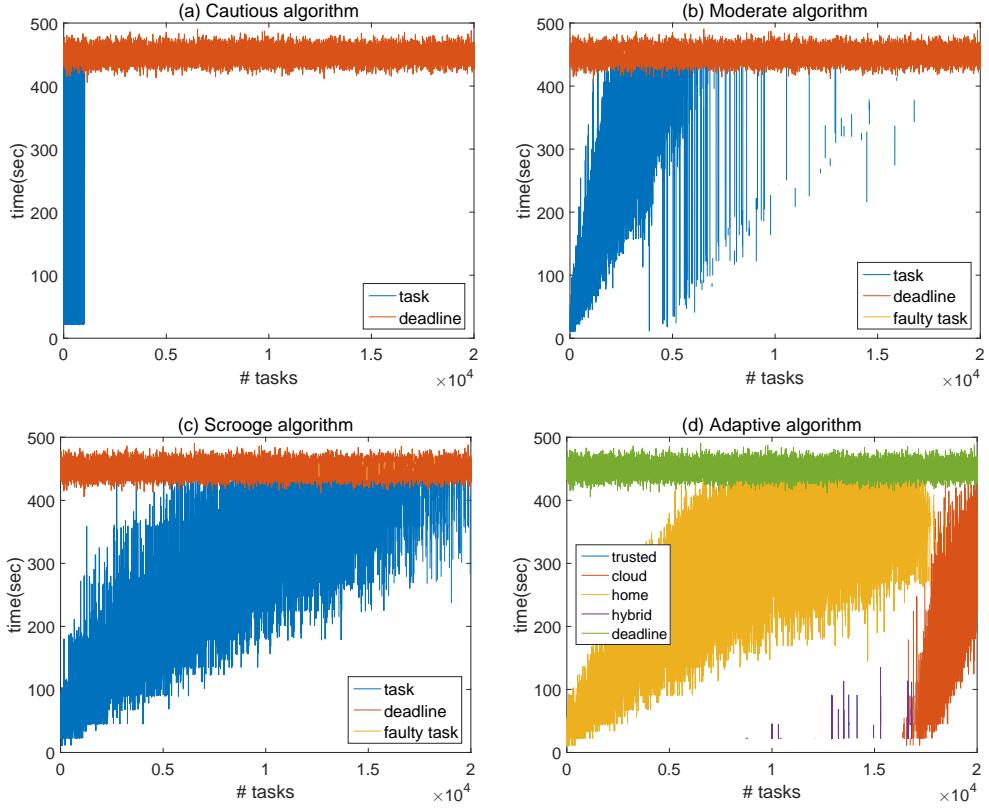


Figure 3: Computation time of tasks with high load

- middleware for byzantine fault tolerant database replication. In *Proc. of the 6th ACM Conf. on Computer Systems*, pages 107–122, 2011.
- [18] K. Hoffman, D. Zage, and C. Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Comput. Surv.*, 42(1):1:1–1:31, Dec. 2009.
- [19] Z. Hong and Z. Hai. Failure-aware resource scheduling policy for hybrid cloud. In *Int. Conf. on Computational Intelligence and Security*, pages 152–156, 2015.
- [20] B. Javadi, J. Abawajy, and R. Buyya. Failure-aware resource provisioning for hybrid cloud infrastructure. *J. Parallel Distrib. Comput.*, 72(10):1318–1331, 2012.
- [21] B. M. Kapron, D. Kempe, V. King, J. Saia, and V. Sanwalani. Fast asynchronous byzantine agreement and leader election with full information. *ACM Trans. Algorithms*, 6(4):68:1–68:28, 2010.
- [22] M. Mitzenmacher. The power of two choices in randomized load balancing. *IEEE Trans. Parallel Distrib. Syst.*, 12(10):1094–1104, Oct. 2001.
- [23] M. Moca, C. Litan, G. C. Silaghi, and G. Fedak. Multi-criteria and satisfaction oriented scheduling for hybrid distributed computing infrastructures. *Future Generation Computer Systems*, 55:428–443, Feb. 2016.
- [24] A. Mostefaoui, H. Moumen, and M. Raynal. Signature-free asynchronous byzantine consensus with  $t < n/3$  and  $o(n^2)$  messages. In *Proc. of the ACM SOSP*, pages 2–9, 2014.
- [25] M. O. Rabin. Randomized byzantine generals. In *Proc.*

- of the 24th Annual Symp. on Foundations of Computer Science (FOCS)*, pages 403–409, 1983.
- [26] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman. Reputation systems. *Commun. ACM*, 43(12):45–48, 2000.
- [27] B. Rood and M. J. Lewis. Multi-state grid resource availability characterization. In *Proc. of the IEEE/ACM Int. Conf. on Grid Computing*, pages 42–49, 2007.
- [28] L. Sarmenta. Sabotage-Tolerance Mechanisms for Volunteer Computing Systems. *Future Generation Computing Systems*, 14(4):561–572, 2002.
- [29] J. D. Sonnek, M. Nathan, A. Chandra, and J. B. Weissman. Reputation-based scheduling on unreliable distributed infrastructures. In *26th IEEE ICDCS*, page 30, 2006.
- [30] S. Toueg. Randomized byzantine agreements. In *Proc. of the ACM PODC*, pages 163–178, 1984.
- [31] G. S. Veronese, M. Correia, A. N. Bessani, L. C. Lung, and P. Verissimo. Efficient byzantine fault-tolerance. *IEEE Trans. on Computers*, 62(1):16–30, 2013.
- [32] J. Yin, J.-P. Martin, A. Venkataramani, L. Alvisi, and M. Dahlin. Separating Agreement from Execution for Byzantine Fault Tolerant Services. In *Proc. of the ACM SOSP*, pages 253–267, 2003.

# A Self-Stabilizing Minimal k-Grouping Algorithm

Ajoy K. Datta  
 University of Nevada, Las Vegas.  
 ajoy.datta@unlv.edu

Toshimitsu Masuzawa  
 Osaka University.  
 masuzawa@ist.osaka-u.ac.jp

Laurence L. Larmore  
 University of Nevada, Las Vegas.  
 lawrence.larmore@unlv.edu

Yuichi Sudo  
 Osaka University.  
 y-sudou@ist.osaka-u.ac.jp

## ABSTRACT

We consider the minimal  $k$ -grouping problem: given a graph  $G = (V, E)$  and a constant  $k$ , partition  $G$  into subgraphs of diameter no greater than  $k$ , such that the union of any two subgraphs has diameter greater than  $k$ . We give a silent self-stabilizing asynchronous distributed algorithm for this problem in the composite atomicity model of computation, assuming the network has unique process identifiers. Our algorithm works under the *weakly-fair daemon*. The time complexity (*i.e.* the number of rounds to reach a legitimate configuration) of our algorithm is  $O(\frac{nD}{k})$  where  $n$  is the number of processes in the network and  $D$  is the diameter of the network. The space complexity of each process is  $O((n + n_{\text{false}}) \log n)$  where  $n_{\text{false}}$  is the number of false identifiers, *i.e.*, identifiers that do not match the identifier of any process, but which are stored in the local memory of at least one process at the initial configuration. Our algorithm guarantees that the number of groups is at most  $2n/k + 1$  after convergence. We also give a novel composition technique to concatenate a silent algorithm repeatedly, which we call *loop composition*.

## 1. INTRODUCTION

Modern networks or distributed systems generally consist of numerous computers (or processes). Therefore, it is important, in some applications, to partition such a system into a set of groups, among each of which processes communicate with each other without much delay. This problem has been formalized as  $k$ -clustering. In the literature, the following similar but different two definitions exist for  $k$ -clustering: given a graph  $G(V, E)$  and a constant integer  $k$ , one is to find a partition of  $V$  into  $\{V_1, \dots, V_s\}$  such that every subgraph  $G(V_i)$  induced by  $V_i$  has radius no greater than  $k$ , and the other one is to find a partition of  $V$  into  $\{V_1, \dots, V_s\}$  such that every subgraph  $G(V_i)$  has diameter no greater than  $k$ . In the former case, it is also required to

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

*ICDCN '17, January 04 - 07, 2017, Hyderabad, India*

© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007772>

designate one process of each subgraph  $G(V_i)$  as a cluster-head such that all processes of  $G(V_i)$  are located within  $k$  hops from the cluster head. In this paper, we call the former *asymmetric  $k$ -clustering* and call the latter  *$k$ -grouping*.

## Our Contributions

This paper considers  $k$ -grouping. We aim to construct a minimal  $k$ -grouping, since finding the minimum  $k$ -grouping is known to be *NP-hard* for  $k > 0$  [6]. Specifically, a partition of  $V$  into  $V_1, \dots, V_s$  is said to be a minimal  $k$ -grouping if every two distinct subgraphs  $G(V_i)$  and  $G(V_j)$  are unmergeable, that is,  $G(V_i \cup V_j)$  has diameter greater than  $k$ . We give a self-stabilizing minimal  $k$ -grouping algorithm for any undirected graph with unique process-identifiers. The time complexity of our algorithm is  $O(nD/k)$  rounds, where  $n = |V|$  and  $D$  is the diameter of the network, while the space complexity is  $O((n + n_{\text{false}}) \log n)$  bits per process, where  $n_{\text{false}}$  is the number of false identifiers, *i.e.*, identifiers stored in the memory of processes in the initial configuration which do not match the identifier of any process. Our algorithm also guarantees that the number of groups is at most  $2n/k + 1$ .

We introduce a novel composition technique, *loop composition*, to define our algorithm. To the best of our knowledge, this is the first composition technique that enables the same algorithm (called the *base algorithm*) to be executed arbitrarily many times repeatedly. Specifically, every time an execution of the base algorithm terminates, a new execution of the base algorithm starts after the values of all output variables are copied to the corresponding input variables, unless a specific condition holds. This composition technique reuses the same variables of the base algorithm repeatedly. It helps us design an algorithm with small space complexity. Moreover, it also helps us eliminate certain assumptions. For example, our algorithm repeats at most  $n/k$  executions of its base algorithm, but it does not need to know even an upper bound of  $n$ .

## Related Work

Distributed  $k$ -grouping algorithms are given in [11, 10]. Fernandes and Malkhi [11] give a non self-stabilizing  $k$ -grouping algorithm in a unit disk graph. Their algorithm does not guarantee minimality, but does guarantee an  $O(k)$ -approximation, meaning that the number of groups it produces is within an  $O(k)$  factor of the optimum number of groups. Its time complexity is  $O(n)$ , and its space complexity is

$O(k \log n)$  bits per process. Ducourthial and Khalfallah [10] give a self-stabilizing  $k$ -grouping algorithm in a unit disk graph. Their algorithm guarantees minimality of groups. However, no upper bound on the number of groups is given. The upper bound of the time complexity is also not given.

Finding the minimum asymmetric  $k$ -clustering is also  $NP$ -hard [1]. Two self-stabilizing asymmetric  $k$ -clustering algorithms exist in the literature [3, 4]; The algorithm of [3] is  $O(k)$ -competitive while that of [4] guarantees that clusters it produces are minimal and the number of the clusters is at most  $(k + 1)/n$ .

There are a variety of techniques to compose two or more self-stabilizing algorithms in the literature [8, 9, 2, 12, 4, 13, 14], such as fair-composition [9] and parallel composition [8], but none of them enables an unbounded number of repetitions of the same algorithm, such as ours.

## 2. PRELIMINARIES

A connected undirected network  $G = (V, E)$  of  $n = |V|$  processes is given where  $n \geq 2$ . Each process  $v$  has a unique identifier  $v.id$  chosen from a set  $ID$  of non-negative integers. We assume that  $|ID| \leq O(n^c)$  holds for some constant  $c$ , thus, a process can store an identifier in  $O(\log n)$  space. By an abuse of notation, we will identify each process with its identifier, and vice versa, whenever convenient. We call a member of  $ID$  a *false identifier* if it does not match the identifier of any process in  $V$ .

We use the locally shared memory model [7]. A process is modeled by a state machine and its state is defined by the values of its variables. A process can read the values of its own and its neighbors' variables simultaneously, but can update only its own variables. An algorithm of each process  $v$  is defined to be a finite set of actions of the following form:  $<label><guard>\longrightarrow<statement>$ . The *label* of each action is used for reference. The *guard* is a predicate on the variables and identifiers of  $v$  and its neighbors. The *statement* is an assignment which updates the state of  $v$ . An action can be executed only if it is *enabled*, i.e. its guard evaluates to true, and a process is *enabled* if at least one of its actions is enabled. The evaluation of a guard and the execution of the corresponding statement are presumed to take place in one atomic step, according to the composite atomicity model [8].

A *configuration* of the network is a vector consisting of a state for each process. We denote by  $\gamma(v).x$  the value of variable  $x$  of process  $v$  in configuration  $\gamma$ . Each transition from a configuration to another, called a *step* of the algorithm, is driven by a *daemon*. We assume the *distributed daemon* in this paper; at each step, the distributed daemon selects one or more enabled processes to execute an action. If a selected process has two or more enabled actions, it executes the action with the smallest label number. We write  $\gamma \mapsto_{\mathcal{A}} \gamma'$  if configuration  $\gamma$  can change to  $\gamma'$  by one step of algorithm  $\mathcal{A}$ . We define an *execution* of algorithm  $\mathcal{A}$  to be a sequence of configurations  $\gamma_0, \gamma_1, \dots$  such that  $\gamma_i \mapsto_{\mathcal{A}} \gamma_{i+1}$  for all  $i \geq 0$ . We assume the daemon to be *weakly-fair*, meaning that a continuously enabled process must be selected eventually.

An execution is *maximal* if it is infinite, or it terminates at a *final* configuration, i.e. a configuration at which no process is enabled. Let  $\mathcal{L}$  be a predicate on configurations. Algorithm  $\mathcal{A}$  is said to be *self-stabilizing* for  $\mathcal{L}$  if every maximal execution  $\gamma_0, \gamma_1, \dots$  of  $\mathcal{A}$  satisfies both (i)  $\mathcal{L}(\gamma_i)$  for some  $i$ , and (ii)  $\mathcal{L}(\gamma_j) \Rightarrow \mathcal{L}(\gamma_{j'})$  for any  $0 \leq j \leq j'$ . We also

say that  $\mathcal{A}$  is *silent* if every execution of  $\mathcal{A}$  is finite. In the remainder of this paper, we say “ $\gamma \in \mathcal{L}$ ” or “configuration  $\gamma$  in  $\mathcal{L}$ ” to mean that  $\mathcal{L}(\gamma) = \text{true}$ .

We measure time complexity of an execution in *rounds* [8]. We say that process  $v$  is *neutralized* at step  $\gamma_i \mapsto \gamma_{i+1}$  if  $v$  is enabled at  $\gamma_i$  and not at  $\gamma_{i+1}$ . We define the first round of an execution  $\varrho = \gamma_0, \gamma_1, \dots$  to be the minimum prefix  $\gamma_0 \dots \gamma_s$  during which every process enabled at  $\gamma_0$  executes an action or is neutralized. The second round of  $\varrho$  is defined to be the first round of the execution  $\gamma_s, \gamma_{s+1}, \dots$ , and so forth. We measure the *time* of  $\varrho$  to be its number of rounds.

Let  $N_v$  denote the neighbors of a process  $v$ , i.e.  $N_v = \{u \mid \{u, v\} \in E\}$ . The distance  $d(u, v)$  between processes  $u$  and  $v$  is defined to be the smallest length of any path between them, where length is defined to be the number of edges. Define  $N_v^i = \{u \in V \mid d(u, v) \leq i\}$ , the *i*-neighborhood of  $v$ . Note that  $N_v^1 = N_v \cup \{v\}$ . Graph  $G(V')$  denotes the subgraph of  $G = (V, E)$  induced by the set of processes  $V' \subseteq V$ . Define  $d_{V'}$ , or  $d_{G(V')}$ , to be the distance in  $G(V')$ . If there is no path in  $G(V')$  between  $u$  and  $v$ , we say  $d_{V'}(u, v) = \infty$ . We denote the diameter of  $G(V')$  by  $D(G(V'))$ , defined to be  $\max\{d_{V'}(u, v) \mid u, v \in V'\}$ . We write  $D = D(G)$ . (Note that  $D(G') = \infty$  if  $G'$  is not connected.)

By an abuse of notation, we sometimes regard a predicate on configurations as the set of configurations throughout the paper. For example, we write  $\gamma \in \mathcal{L}_1 \cap \mathcal{L}_2$  when a configuration  $\gamma$  satisfies both predicates  $\mathcal{L}_1$  and  $\mathcal{L}_2$ .

## Problem Specification

Given an integer  $k$ , our goal is to find a *minimal* partition<sup>1</sup>  $\{V_1, \dots, V_s\}$  with diameter-bound  $k$ , which means (i)  $V = \bigcup_{i=1}^s V_i$ , (ii)  $V_i \cap V_j = \emptyset$  for any  $i \neq j$ , (iii)  $D(G(V_i)) \leq k$  for any  $i$ , and  $D(G(V_i \cup V_j)) > k$  holds for  $i \neq j$ . We assume that each process  $v$  has variable  $v.group \in ID$ . We define predicate  $\mathcal{L}_k(\gamma)$  as follows:  $\mathcal{L}_k(\gamma) = \text{true}$  if and only if, in configuration  $\gamma$ ,  $\{V(i) \neq \emptyset \mid i \in V\}$  is a minimal partition with diameter-bound  $k$  where  $V(i) = \{u \in V \mid u.group = i\}$ . Our goal is to develop a silent self-stabilizing algorithm for  $\mathcal{L}_k$ .

## 3. LOOP COMPOSITION

In this section, we give a novel composition technique to develop a new algorithm. Given two algorithms  $\mathcal{A}$  and  $\mathcal{P}$  and a predicate  $E$ , this technique generates a silent self-stabilizing algorithm  $\text{Loop}(\mathcal{A}, E, \mathcal{P})$  for predicate  $\mathcal{L}$ , whose time complexity is  $O(n + T_{\mathcal{P}} + L_{\mathcal{A}}(T_{\mathcal{A}} + D))$  rounds. As we shall see later,  $T_{\mathcal{A}}$  and  $T_{\mathcal{P}}$  are the time complexities of  $\mathcal{A}$  and  $\mathcal{P}$  respectively, and  $L_{\mathcal{A}}$  is an upper bound on the number of iterations of  $\mathcal{A}$ 's executions.

### 3.1 Preliminaries

Algorithm  $\mathcal{A}$  is the base algorithm of  $\text{Loop}(\mathcal{A}, E, \mathcal{P})$ , and is executed repeatedly during one execution of  $\text{Loop}(\mathcal{A}, E, \mathcal{P})$ . Algorithm  $\mathcal{A}$  assumes that the network always stays at a configuration satisfying a specific condition. Predicate  $E$  is used to detect that the network deviates from that condition, and algorithm  $\mathcal{P}$  is invoked to initialize the network

<sup>1</sup>This definition of minimality, which is the same as [10], does not allow any pair of groups to be mergeable, but it allows some three or more groups to be mergeable. Intuitively, it seems quite costly (perhaps requires exponential time), even in centralized computing, to find a “strict minimal” solution where no set of groups is mergeable.

when such a deviation is detected. We define  $O_{\mathcal{A}}$  (resp.  $O_{\mathcal{P}}$ ) as the set of variables whose values can be updated by an action of  $\mathcal{A}$  (resp.  $\mathcal{P}$ ), and  $I_{\mathcal{A}}$  (resp.  $I_{\mathcal{P}}$ ) as the set of variables whose value is never updated and only read by an action of  $\mathcal{A}$  (resp.  $\mathcal{P}$ ). We assume  $O_{\mathcal{A}} \cap O_{\mathcal{P}} = \emptyset$  and  $I_{\mathcal{P}} = \emptyset$ . The error detecting predicate  $E(v)$  is evaluated by process  $v \in V$ , and its value depends only on variables of  $I_{\mathcal{A}} \cup O_{\mathcal{P}}$  of the processes of  $N_v^1$ . Let  $\mathcal{E}$  be a predicate on configurations such that  $\mathcal{E}(\gamma)$  holds if and only if  $\bigvee_{v \in V} E(v)$  holds in configuration  $\gamma$ . We assume that algorithm  $\mathcal{A}$  has a *copying variable*  $\bar{x} \in I_{\mathcal{A}}$  for every variable  $x \in O_{\mathcal{A}}$ . We define  $\gamma^{\text{copy}}$  as the configuration obtained by replacing the value of  $v.\bar{x}$  with the value of  $v.x$  for every process  $v$  and every variable  $x \in O_{\mathcal{A}}$  in configuration  $\gamma$ . We define predicate  $\mathcal{C}_{\text{goal}}(\mathcal{A}, E)$  as follows: configuration  $\gamma$  satisfies  $\mathcal{C}_{\text{goal}}(\mathcal{A}, E)$  if and only if  $\gamma \in \neg\mathcal{E}$ ,  $\gamma^{\text{copy}} = \gamma$ , and no action of  $\mathcal{A}$  is enabled in any process. In the rest of this section, we simply denote  $\mathcal{C}_{\text{goal}}(\mathcal{A}, E)$  by  $\mathcal{C}_{\text{goal}}$ . We assume that  $\mathcal{A}$  satisfies the following three requirements: every maximal execution of  $\mathcal{A}$  that starts from a configuration in  $\neg\mathcal{E}$  terminates, within  $T_{\mathcal{A}}$  rounds, at a configuration  $\gamma$  such that  $\gamma^{\text{copy}} \in \neg\mathcal{E}$  (**Shiftable Convergence**), if  $\varrho_0, \varrho_1 \dots$  is an infinite sequence of maximal executions of  $\mathcal{A}$  where  $\varrho_i = \gamma_{i,0}, \gamma_{i,1}, \dots, \gamma_{i,s_i}$ ,  $\gamma_{0,0} \in \neg\mathcal{E}$ , and  $\gamma_{i+1,0} = \gamma_{i,s_i}^{\text{copy}}$  for each  $i \geq 0$ , then  $\gamma_{j,s_j} \in \mathcal{C}_{\text{goal}}$  holds for some  $j < L_{\mathcal{A}}$  (**Loop Convergence**), and  $\gamma \in \mathcal{C}_{\text{goal}} \Rightarrow \gamma \in \mathcal{L}$  holds for every configuration  $\gamma$  (**Correctness**). Algorithm  $\mathcal{P}$  is used to initialize a network when the network stays at a configuration in  $\mathcal{E}$ . We assume that every maximal execution of  $\mathcal{P}$  terminates at a configuration in  $\neg\mathcal{E}$  within  $T_{\mathcal{P}}$  rounds.

### 3.2 Algorithm Loop( $\mathcal{A}, E, \mathcal{P}$ )

In this subsection, we present an algorithm Loop( $\mathcal{A}, E, \mathcal{P}$ ) given  $\mathcal{A}$ ,  $E$ , and  $\mathcal{P}$  satisfying the above requirements. Our algorithm is a silent self-stabilizing for  $\mathcal{L}$  and its time complexity is  $O(n + T_{\mathcal{P}} + L_{\mathcal{A}}(T_{\mathcal{A}} + D))$  rounds.

Our strategy to implement Loop( $\mathcal{A}, E, \mathcal{P}$ ) is simple: The network executes  $\mathcal{A}$  repeatedly while it stays at configurations in  $\neg\mathcal{E}$ . When an execution of  $\mathcal{A}$  terminates, each process  $v$  copies the value of  $v.x$  to  $v.\bar{x}$  for all  $x \in O_{\mathcal{A}}$ , and the network restarts a new execution of  $\mathcal{A}$ , unless it reaches a configuration in  $\mathcal{C}_{\text{goal}}$ . The network executes  $\mathcal{P}$  when it stays in  $\mathcal{E}$ . The assumptions of  $\mathcal{A}$  and  $\mathcal{P}$  guarantee that the network eventually reaches a configuration in  $\mathcal{L}$ .

We give the actions of Loop( $\mathcal{A}, E, \mathcal{P}$ ) in Table 1. In what follows, we denote by  $\mathcal{X}$ -Enabled( $v$ ) that some action of algorithm  $\mathcal{X}$  is enabled at process  $v$ .

We use the algorithm of [5], (denoted by BFS in this paper) as a module to construct a BFS tree ( $L_1$ ). The tree is used as a communication backbone for Loop( $\mathcal{A}, E, \mathcal{P}$ ). Algorithm BFS is silent and self-stabilizing and constructs a BFS tree: each process  $v$  has variable  $v.\text{parent} \in N_v \cup \{\perp\}$  where  $\perp$  means *undefined* or *null value*, and every maximal execution of BFS terminates at a configuration in  $\mathcal{L}_{\text{BFS}}$  where  $r.\text{parent} = \perp$  for some  $r \in V$  and the set of edges  $\{(v, v.\text{parent}) \mid v \neq r\}$  spans a BFS tree rooted at  $r$ . We define  $\text{Par}(v) = \{v.\text{parent}\}$  and  $\text{Chi}(v) = \{u \in V \mid u.\text{parent} = v\}$ . We regard  $\{\perp\}$  as the empty set, hence  $\text{Par}(r) = \emptyset$  in a configuration satisfying  $\mathcal{L}_{\text{BFS}}$ . Thus, a predicate such as “ $\forall u \in \text{Par}(r) : \dots$ ” always holds, and a predicate such as “ $\exists u \in \text{Par}(r) : \dots$ ” never holds. This algorithm terminates within  $O(n)$  rounds and uses  $O(\log n)$  space per process. We discuss the following part assuming that an execution of BFS already terminated and the net-

work stays at configurations in  $\mathcal{L}_{\text{BFS}}$ .

The difficulty we face is to detect termination of each execution of  $\mathcal{A}$  and  $\mathcal{P}$ , and prevent any two processes from performing different executions at the same time (*e.g.* executions of  $\mathcal{A}$  and  $\mathcal{P}$  or the  $i^{\text{th}}$  execution and  $(i+1)^{\text{st}}$  execution of  $\mathcal{A}$ ). We overcome it with a *color wave* mechanism. Specifically, each process  $v$  has three variables  $v.\text{cl} \in \{0, 1, 2, 3, 4\}$ ,  $v.\text{mode} \in \{A, P\}$  and  $v.\text{rst} \in \{0, 1\}$ . Variable  $v.\text{cl}$  is the *color* of process  $v$ . Colors 0, 1, and 2 mean that the current execution of  $\mathcal{A}$  or  $\mathcal{P}$  may not have terminated yet, Color 3 means that the execution has already terminated, and Color 4 means that the network is now in transition to the next execution. Variable  $v.\text{mode}$  indicates which algorithm is currently executing. As we shall see later, all processes must have the same mode unless some process has color 4. Process  $v$  executes  $\mathcal{A}$  (resp.  $\mathcal{P}$ ) if  $\mathcal{A}$ -Enabled( $v$ ) (resp.  $\mathcal{P}$ -Enabled( $v$ )) and every  $u \in N_v^1$  satisfies  $u.\text{mode} = A$  (resp.  $u.\text{mode} = P$ ) and  $u.\text{cl} \neq 4$  ( $L_7$ - $L_8$ ). A *reset flag* is used to prevent any process from having color 3 or 4 until the current execution of  $\mathcal{A}$  or  $\mathcal{P}$  terminates. Process  $v$  raises a reset flag (*i.e.*  $v.\text{rst} \leftarrow 1$ ) every time it executes an action of  $\mathcal{A}$  or  $\mathcal{P}$  ( $L_7$ - $L_8$ ). The raised flag is propagated from  $v$  to the root  $r$  through the BFS tree ( $L_{10}$ ), resulting in changing  $r$ 's color to 0 unless it is already 3 or 4 ( $L_2$ ). The reset flags are eventually dropped in order from  $v$  to  $r$  ( $L_{11}$ ). Once  $r$ 's color is reset to 0, all other processes will also change their colors to 0 because a process changes its color to 0 when its parent has color 0 ( $L_3$ - $L_4$ ).

Color waves are propagated through the BFS tree (Figure 1). Suppose that all processes have color 0 now. When  $r.\text{rst} = 0$ , the root  $r$  begins a top-down color-1-wave changing all processes' colors from 0 to 1. Specifically, each  $v \in V$  changes its color from 0 to 1 when its parent has color 1 ( $r$  ignores this condition), all its children have color 0, and  $v.\text{rst} = 0$  ( $L_{12}$ ). After the color-1-wave reaches all leaves, a bottom-up color-2-wave begins from leaves to  $r$  changing all processes' color from 1 to 2. Specifically, each  $v \in V$  changes its color from 1 to 2 when its parent has color 1 ( $r$  ignores this condition), and all its children have color 2 ( $L_{13}$ ). As we will prove later, the current execution of  $\mathcal{A}$  or  $\mathcal{P}$  has already terminated when  $r$  receives color-2-wave (Lemma 8). Thereafter,  $r$  begins a top-down color-3-wave in the same way as a color-1-wave ( $L_{12}$ ). A process changes its color from 3 to 4 if the network should shift to the next execution of  $\mathcal{A}$ . Specifically, process  $v$  with mode  $A$  changes its color to 4 if some process has  $\bar{x} \neq x$  for some variable  $x \in O_{\mathcal{A}}$ , and it performs  $v.\bar{x} \leftarrow v.x$  for all variables  $x \in O_{\mathcal{A}}$  for the next execution of  $\mathcal{A}$  ( $L_{14}$ ). Note that this color-4-wave is propagated by simple flooding (not through the BFS tree). A process with mode  $P$  changes its color to 4 without this condition, and changes its mode to  $A$  ( $L_{15}$ ). In both cases, a process changes its color to 4 only after all its children change their colors from 2 to 3 ( $L_{14}$ ,  $L_{15}$ ); Otherwise, a color-3-wave may stop at the process. Finally, a bottom-up Color-0-wave moves from the leaves to  $r$  changing all colors from 4 to 0. Specifically, a process changes its color from 4 to 0 if its parent has color 4 ( $r$  ignores this condition), all its children have color 0, and every neighboring process has color 0 or 4 ( $L_{16}$ ). The last condition is needed to prevent the process from executing the next execution of  $\mathcal{A}$  before all its neighbors finish their copy procedure ( $v.\bar{x} \leftarrow v.x$ ). All processes eventually return to color 0, and the network starts a new execution of  $\mathcal{A}$ .

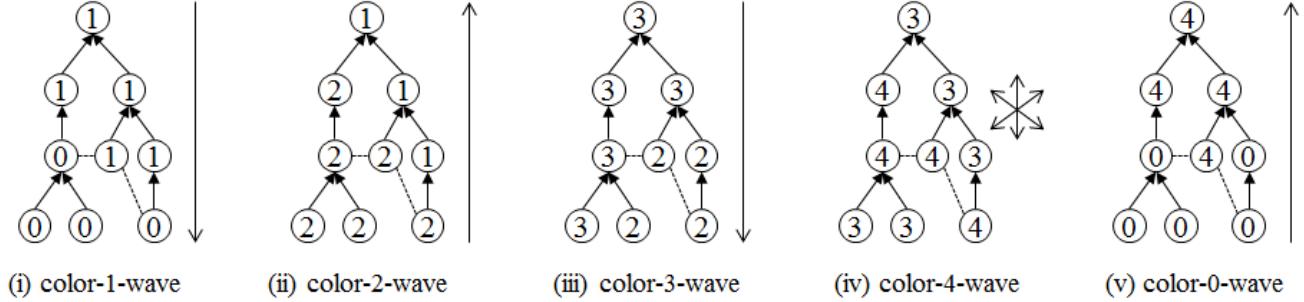
**Table 1:** Loop( $\mathcal{A}, E, \mathcal{P}$ )

[Notations]

$$\begin{aligned} \text{Illegal\_Pair} &= \{(1, 3), (1, 4), (2, 0), (2, 1), (2, 3), (2, 4), (3, 0), (3, 1), (4, 1), (4, 2)\} \\ \text{Down\_OK}(v) &\equiv (\forall u \in \text{Par}(v) : u.\text{cl} = v.\text{cl} + 1) \wedge (\forall u \in \text{Chi}(v) : u.\text{cl} = v.\text{cl}) \\ \text{Up\_OK}(v) &\equiv (\forall u \in \text{Par}(v) : u.\text{cl} = v.\text{cl}) \wedge (\forall u \in \text{Chi}(v) : u.\text{cl} \equiv v.\text{cl} + 1 \pmod{5}) \end{aligned}$$

[Actions of process  $v$ ]

L <sub>1</sub>	BFS-Enabled( $v$ )	→ execute BFS
L <sub>2</sub>	( $v.\text{cl} \notin \{0, 3, 4\}$ ) $\wedge$ ( $v.\text{rst} = 1$ )	→ $v.\text{cl} \leftarrow 0$
L <sub>3</sub>	( $v.\text{cl} \in \{1, 2\}$ ) $\wedge$ ( $\exists u \in \text{Par}(v) : u.\text{cl} = 0$ )	→ $v.\text{cl} \leftarrow 0$
L <sub>4</sub>	( $v.\text{cl} \in \{3, 4\}$ ) $\wedge$ ( $\exists u \in \text{Par}(v) : u.\text{cl} = 0$ )	→ $v.\text{cl} \leftarrow 0, v.\text{rst} \leftarrow 1$
L <sub>5</sub>	( $v.\text{mode} = A$ ) $\wedge$ $E(v)$ $\wedge$ ( $\forall u \in N_v^1 : u.\text{cl} \neq 4$ )	→ $v.\text{mode} \leftarrow P, v.\text{rst} \leftarrow 1$
L <sub>6</sub>	( $v.\text{mode} = A$ ) $\wedge$ ( $v.\text{cl} \neq 4$ ) $\wedge$ ( $\exists u \in N_v : u.\text{mode} = P$ )	→ $v.\text{mode} \leftarrow P, v.\text{rst} \leftarrow 1$
L <sub>7</sub>	$\mathcal{A}$ -Enabled( $v$ ) $\wedge$ ( $\forall u \in N_v^1 : u.\text{mode} = A \wedge u.\text{cl} \neq 4$ )	→ execute $\mathcal{A}$ , $v.\text{rst} \leftarrow 1$
L <sub>8</sub>	$\mathcal{P}$ -Enabled( $v$ ) $\wedge$ ( $\forall u \in N_v^1 : u.\text{mode} = P \wedge u.\text{cl} \neq 4$ )	→ execute $\mathcal{P}$ , $v.\text{rst} \leftarrow 1$
L <sub>9</sub>	$\exists u \in \text{Chi}(v) : (v.\text{cl}, u.\text{cl}) \in \text{Illegal\_Pair}$	→ $v.\text{cl} \leftarrow 0, v.\text{rst} \leftarrow 1$
L <sub>10</sub>	( $v.\text{rst} = 0$ ) $\wedge$ ( $\exists u \in \text{Chi}(v) : u.\text{rst} = 1$ )	→ $v.\text{rst} \leftarrow 1$
L <sub>11</sub>	( $v.\text{rst} = 1$ ) $\wedge$ ( $\forall u \in \text{Par}(v) : u.\text{rst} = 1$ )	→ $v.\text{rst} \leftarrow 0$
	$\wedge (\forall u \in \text{Chi}(v) : u.\text{rst} = 0)$	
L <sub>12</sub>	( $v.\text{cl} \in \{0, 2\}$ ) $\wedge$ Down_OK( $v$ ) $\wedge$ ( $v.\text{rst} = 0$ )	→ $v.\text{cl} \leftarrow v.\text{cl} + 1$
L <sub>13</sub>	( $v.\text{cl} = 1$ ) $\wedge$ Up_OK( $v$ )	→ $v.\text{cl} \leftarrow 2$
L <sub>14</sub>	( $v.\text{cl} = 3$ ) $\wedge$ ( $v.\text{mode} = A$ ) $\wedge$ ( $\forall u \in \text{Chi}(v) : u.\text{cl} \neq 2$ )	→ $v.\text{cl} \leftarrow 4, v.\bar{x} \leftarrow v.x$
	$\wedge (\exists x \in O_{\mathcal{A}} : v.\bar{x} \neq v.x \vee \exists u \in N_v : u.\text{cl} = 4)$	for all $x \in O_{\mathcal{A}}$
L <sub>15</sub>	( $v.\text{cl} = 3$ ) $\wedge$ ( $v.\text{mode} = P$ ) $\wedge$ ( $\forall u \in \text{Chi}(v) : u.\text{cl} \neq 2$ )	→ $v.\text{cl} \leftarrow 4, v.\text{mode} \leftarrow A$
L <sub>16</sub>	( $v.\text{cl} = 4$ ) $\wedge$ Up_OK( $v$ ) $\wedge$ ( $\forall u \in N_v : u.\text{cl} \in \{0, 4\}$ )	→ $v.\text{cl} \leftarrow 0$



**Figure 1:** Color-waves in Loop( $\mathcal{A}, E, \mathcal{P}$ ). Numbers in circles represent colors of processes. Arrows represent edges of the BFS tree, and dashed lines represent other edges.

The states of the processes may be incoherent in an arbitrary initial configuration, but Loop( $\mathcal{A}, E, \mathcal{P}$ ) resolves the incoherence. Process  $v$  changes its mode from  $A$  to  $P$  when  $v$  finds  $E(v) = \text{true}$  or some neighboring process has mode  $P$  (L<sub>5</sub>-L<sub>6</sub>). At this time,  $v$  ignores  $E(v) = \text{true}$  when some process  $u \in N_v^1$  has color 4 (L<sub>5</sub>), and ignores a neighboring process with mode  $P$  when  $v.\text{cl} = 4$  (L<sub>6</sub>). These exceptions are needed because, during a color-0-wave,  $E(v)$  cannot be evaluated correctly and a process with mode  $A$  can have a neighbor with mode  $P$ . Color-incoherence preventing color waves is removed as follows: process  $v$  changes its color to 0 if some  $u \in \text{Chi}(v)$  satisfies

$$(v.\text{cl}, u.\text{cl}) \in \text{Illegal\_Pair} = \{(1, 3), (1, 4), (2, 0), (2, 1), (2, 3), (2, 4), (3, 0), (3, 1), (4, 1), (4, 2)\}.$$

A reset flag is raised every time incoherence is detected and solved (L<sub>4</sub>, L<sub>5</sub>, L<sub>6</sub>, or L<sub>9</sub>) except for L<sub>3</sub>. This exception exists only to simplify the proof, as we shall see later.

An execution of Loop( $\mathcal{A}, E, \mathcal{P}$ ) terminates when every pro-

cess  $v$  satisfies  $v.\text{mode} = A$ ,  $v.\text{cl} = 3$ ,  $v.\text{rst} = 0$ ,  $E(v) = \text{false}$ ,  $\mathcal{A}$ -Enabled( $v$ ) =  $\text{false}$ , and  $v.x = v.\bar{x}$  for all  $x \in O_{\mathcal{A}}$ . We denote the set of such configurations by  $\mathcal{C}_{\text{fin}}$ . Note that  $\mathcal{C}_{\text{fin}} \subseteq \mathcal{C}_{\text{goal}} \subseteq \mathcal{L}$ .

### 3.3 Correctness

In this subsection, we prove that every maximal execution of Loop( $\mathcal{A}, E, \mathcal{P}$ ) terminates at a configuration in  $\mathcal{C}_{\text{fin}}$  within  $O(n + T_{\mathcal{P}} + L_{\mathcal{A}}(T_{\mathcal{A}} + D))$  rounds regardless of its initial configuration. For  $i = 0, 1, 2, 3, 4$ , we define  $\mathcal{R}_i$  as the set of configurations in  $\mathcal{L}_{\text{BFS}}$  where the root  $r$  has color  $i$ . We define  $\mathcal{C}_S$  as the set of configurations in  $\mathcal{L}_{\text{BFS}}$  where every process has a color in  $S \subseteq \{0, 1, 2, 3, 4\}$ .

**Lemma 1** Let  $\varrho = \gamma_0, \gamma_1, \dots$  be a maximal execution of Loop( $\mathcal{A}, E, \mathcal{P}$ ) starting from  $\gamma_0 \in \mathcal{L}_{\text{BFS}}$ . Then,  $\varrho$  reaches a configuration in  $\mathcal{R}_0$  or  $\mathcal{C}_{\text{fin}}$  within  $O(D)$  rounds.

PROOF. Obtained from the following Lemmas 2, 3, 5, 6, and 7. □

**Lemma 2** Let  $\varrho = \gamma_0, \gamma_1, \dots$  be a maximal execution of  $\text{Loop}(\mathcal{A}, E, \mathcal{P})$  starting from  $\gamma_0 \in \mathcal{R}_1$ . Then,  $\varrho$  reaches a configuration in  $\mathcal{R}_0$  or  $\mathcal{R}_2$  within  $O(D)$  rounds.

PROOF. We assume that no process raises a reset flag until  $r$ 's color changes to 0 or 2. This assumption does not lose generality since  $r$ 's color changes to 0 within  $O(D)$  rounds after some process raises a reset flag. By L<sub>4</sub> and L<sub>9</sub>, this assumption guarantees  $\gamma_0 \in \mathcal{C}_{\{0,1,2\}}$  and  $\gamma_0(u).\text{c1} = 2 \Rightarrow \gamma_0(v).\text{c1} = 2$  for all  $u, v \in V$  such that  $v \in \text{Chi}(u)$ . Note that if a process with color 0 has a child with color 1 or 2, the child changes its color to 0 within one round by L<sub>3</sub>. Hence, all processes with color 0 change their colors to 1 within  $O(D)$  rounds by a color-1-wave (L<sub>12</sub>), and all processes with color 1 including  $r$  change their colors to 2 within the next  $O(D)$  rounds by a color-2-wave (L<sub>13</sub>).  $\square$

**Lemma 3** Let  $\varrho = \gamma_0, \gamma_1, \dots$  be a maximal execution of  $\text{Loop}(\mathcal{A}, E, \mathcal{P})$  starting from  $\gamma_0 \in \mathcal{R}_2$ . Then,  $\varrho$  reaches a configuration in  $\mathcal{R}_0$  or  $\mathcal{R}_3$  within  $O(1)$  rounds.

PROOF. The root  $r$  changes its color to 0 by L<sub>9</sub> if it has a child of color 0, 1, 3, or 4. If all children of  $r$  have color 2,  $r$  changes its color to 3. We do not consider a reset flag above, but it just changes  $r$ 's color to 0.  $\square$

**Lemma 4** Let  $\gamma$  be a configuration in  $\mathcal{L}_{\text{BFS}}$  where  $u \in V$  has color 3 and  $v \in \text{Chi}(u)$  has color 2. The color of  $v$  changes to 0 or 3 within  $O(1)$  rounds in every maximal execution  $\varrho$  of  $\text{Loop}(\mathcal{A}, E, \mathcal{P})$  starting from  $\gamma$ .

PROOF. Process  $u$  has color 0 or 3 as long as  $v$  has color 2. Hence,  $v$  always stays enabled unless  $v$  executes some action. Therefore,  $v$  is selected by the scheduler in the first round. At this time,  $v$  changes its color to 0 by L<sub>2</sub>, L<sub>3</sub>, or L<sub>9</sub>, changes its color to 3 by L<sub>12</sub>, or raises a reset flag by L<sub>5</sub>, L<sub>6</sub>, L<sub>7</sub>, L<sub>8</sub>, L<sub>10</sub>. In the last case,  $v$  changes its color to 0 by L<sub>2</sub> the next time the scheduler selects  $v$ .  $\square$

**Lemma 5** Let  $\varrho = \gamma_0, \gamma_1, \dots$  be a maximal execution of  $\text{Loop}(\mathcal{A}, E, \mathcal{P})$  starting from  $\gamma_0 \in \mathcal{R}_3$ . Then,  $\varrho$  reaches a configuration in  $\mathcal{R}_0$ ,  $\mathcal{R}_4$ , or  $\mathcal{C}_{\text{fin}}$  within  $O(D)$  rounds.

PROOF. A process with color 3 changes its color to 0 by L<sub>9</sub> within  $O(1)$  rounds if it has a child with color 0 or 1. A process  $v$  with color 3 changes its color to 0 or 4 by L<sub>9</sub>, L<sub>14</sub>, or L<sub>15</sub> within  $O(1)$  rounds if it has a child with color 4 because all its children with color 2 change their colors to 0 or 3 within  $O(1)$  rounds (Lemma 4). Thus, the network reaches a configuration  $\gamma_i$  in  $\mathcal{R}_0$ ,  $\mathcal{R}_4$ , or  $\mathcal{C}_{\{3\}}$  within  $O(D)$  rounds. When  $\gamma_i \in \mathcal{C}_{\{3\}} \setminus \mathcal{C}_{\text{fin}}$ , some process's color changes to 4 within  $O(1)$  rounds, which changes  $r$ 's color to 4 within  $O(D)$  rounds.  $\square$

**Lemma 6** Let  $\varrho = \gamma_0, \gamma_1, \dots$  be a maximal execution of  $\text{Loop}(\mathcal{A}, E, \mathcal{P})$  starting from  $\gamma_0 \in \mathcal{R}_4$ . Then,  $\varrho$  reaches a configuration in  $\mathcal{R}_4 \cap \mathcal{C}_{\{0,4\}}$  or  $\mathcal{R}_0$  within  $O(D)$  rounds.

PROOF. Lemma 4 and (4, 2)  $\in$  Illegal\_Pair guarantee that, within  $O(D)$  rounds, color 2 disappears from all processes as long as  $r.\text{c1} = 4$ . After that, every process whose parent has color 0 or 4 also has color 0 or 4 within  $O(1)$  rounds and thereafter never has color 1, 2, or 3 as long as  $r.\text{c1} = 4$ . Hence, the lemma is proven by induction on levels (*i.e.* the distance from the root  $r$ ) of processes in the BFS tree.  $\square$

**Lemma 7** Let  $\varrho = \gamma_0, \gamma_1, \dots$  be a maximal execution of  $\text{Loop}(\mathcal{A}, E, \mathcal{P})$  starting from  $\gamma_0 \in \mathcal{R}_4 \cap \mathcal{C}_{\{0,4\}}$ . Then,  $\varrho$  reaches a configuration in  $\mathcal{R}_0$  within  $O(D)$  rounds.

PROOF. No process changes its color to 1, 2, or 3 in a configuration of  $\mathcal{R}_4 \cap \mathcal{C}_{\{0,4\}}$ . Hence, a color-0-wave by L<sub>16</sub> (or a reset flag) changes  $r$ 's color to 0 within  $O(D)$  rounds.  $\square$

**Lemma 8** Let  $\varrho = \gamma_0, \gamma_1, \dots$  be a maximal execution of  $\text{Loop}(\mathcal{A}, E, \mathcal{P})$  where  $\gamma_0 \in \mathcal{R}_0$ . If  $i > 0$  is the smallest integer such that  $\gamma_i \in \mathcal{R}_2$ , then every process  $v$  has the same mode  $\mathcal{X}$  ( $\mathcal{A}$  or  $\mathcal{P}$ ) and satisfies  $\neg \mathcal{X}\text{-Enabled}(v)$ ,  $v.\text{c1} = 2$  and  $v.\text{rst} = 0$  at configuration  $\gamma_i$ .

PROOF. Let  $j$  ( $0 < j < i$ ) be the largest integer such that  $\gamma_{j-1}(r).\text{c1} = 0$  and  $\gamma_j(r).\text{c1} = 1$ . A color-1-wave beginning at  $\gamma_j$  reaches every leaf, and thereafter a color-2-wave is initiated at leaves and reaches  $r$  in  $\gamma_j, \dots, \gamma_i$ . No process has color 3 or 4 in  $\gamma_j$ ; Otherwise, such process  $v$  changes its color from 3 or 4 to 0 and raises a reset flag before  $v$  receives a color-1-wave, which changes  $r$ 's color to 0 again; This is a contradiction to  $j$ 's definition. Since no process changes its color to 3 during  $\gamma_j, \dots, \gamma_i$ , every process has color 0, 1, or 2 during  $\gamma_j, \dots, \gamma_i$ . In the same way, we can prove that all processes have the same mode  $\mathcal{X}$  ( $\mathcal{A}$  or  $\mathcal{P}$ ), and  $\neg \mathcal{X}\text{-Enabled}(v)$  and  $v.\text{rst} = 0$  hold during  $\gamma_j, \dots, \gamma_i$ , hence no process changes its color from 1 or 2 to 0 in the mean time. Thus, every process  $v$  has the same mode  $\mathcal{X}$  ( $\mathcal{A}$  or  $\mathcal{P}$ ) and satisfies  $\neg \mathcal{X}\text{-Enabled}(v)$ ,  $v.\text{c1} = 2$  and  $v.\text{rst} = 0$  at configuration  $\gamma_i$ .  $\square$

We define  $\mathcal{S}_{\mathcal{P}}$  to be the set of configurations in  $\mathcal{R}_0 \cap \mathcal{C}_{\{0,1,2\}}$  where every process has mode  $P$ , and define  $\mathcal{S}_{\mathcal{A}}$  to be the set of configurations in  $\mathcal{R}_0 \cap \mathcal{C}_{\{0,1,2\}}$  and  $\neg \mathcal{E}$  such that every process has mode  $A$ .

**Lemma 9** Let  $\varrho = \gamma_0, \gamma_1, \dots$  be a maximal execution of  $\text{Loop}(\mathcal{A}, E, \mathcal{P})$  starting from  $\gamma_0 \in \mathcal{R}_0$ . Then,  $\varrho$  reaches a configuration in  $\mathcal{S}_{\mathcal{A}}$  or  $\mathcal{S}_{\mathcal{P}}$  within  $O(D)$  rounds.

PROOF. Since  $\gamma_0(r).\text{c1} = 0$ , every process with color 3 or 4 changes its color to 0 within  $O(D)$  rounds. At this time, each such process raises a reset flag by L<sub>4</sub>–L<sub>10</sub>, which will change  $r$ 's color to 0. Thus, the network reaches a configuration  $\gamma$  in  $\mathcal{R}_0 \cap \mathcal{C}_{\{0,1,2\}}$  within  $O(D)$  rounds. If two processes have different modes or some process  $v$  has  $(v.\text{mode} = A) \wedge (E(v) = \text{true})$  in  $\gamma$ , then all processes with mode  $A$  change their modes to  $P$  by L<sub>5</sub> and L<sub>6</sub> within  $O(D)$  rounds, keeping  $r$ 's color 0 with a reset flag, which proves the lemma.  $\square$

**Lemma 10** Let  $\varrho = \gamma_0, \gamma_1, \dots$  be a maximal execution of  $\text{Loop}(\mathcal{A}, E, \mathcal{P})$  starting from  $\gamma_0 \in \mathcal{S}_{\mathcal{P}}$ . Then,  $\varrho$  reaches a configuration in  $\mathcal{S}_{\mathcal{A}}$  within  $O(T_{\mathcal{P}} + D)$  rounds.

PROOF. Every  $\mathcal{P}$ -Enabled process performs  $\mathcal{P}$ 's action by L<sub>8</sub> at least once with every one or two rounds until  $r$ 's color changes to 2. Lemma 8 guarantees that  $r$ 's color never changes to 0 until an execution of  $\mathcal{P}$  terminates. Hence, the network reaches within  $O(T_{\mathcal{P}})$  rounds a configuration in  $\neg \mathcal{E}$  where no process is  $\mathcal{P}$ -Enabled. Thereafter, all reset flags are dropped within  $O(D)$  rounds by L<sub>11</sub>, and  $r$ 's color is 0 at the resulting configuration. The network reaches a configuration in  $\mathcal{C}_{\{2\}}$  and again reaches a configuration  $\gamma \in \mathcal{R}_0$

**Table 2: Non-copy Variables used in *Init* and *Merge***

[Non-array variables]	$v.\text{domain} \in 2^{ID},$ $v.\text{height} \in [0, \lfloor k/2 \rfloor], v.\text{initGroup}, v.\text{group} \in ID$
[Array variables ( $u \in v.\text{domain}$ )] $v.\text{dist}[u], v.\text{groupD}[u], v.\text{mergeD}[u], v.\text{stampD}[u] \in [0, 2k] \cup \{\perp\},$ $v.\text{border}[u], v.\text{far}[u], v.\text{target}[u], v.\text{stamp1}[u], v.\text{stamp2}[u], v.\text{groups}[u] \in ID \cup \{\perp\},$ $v.\text{merging}[u], v.\text{stampON}[u], v.\text{prior}[u] \in \{\text{false}, \text{true}\}$	

within the next  $O(D)$  rounds by  $L_3$ ,  $L_{12}$  (a color-1-wave, a color-3-wave),  $L_{13}$  (a color-2-wave),  $L_{15}$  (a color-4-wave), and  $L_{16}$  (a color-0-wave). Lemma 8 guarantees  $\gamma \in \mathcal{S}_A$ .  $\square$

**Lemma 11** *Let  $\varrho = \gamma_0, \gamma_1, \dots$  be a maximal execution of  $\text{Loop}(A, E, \mathcal{P})$  starting from  $\gamma_0 \in \mathcal{S}_A$ . Then,  $\varrho$  reaches a configuration in  $\mathcal{C}_{\text{fin}}$  within  $O(L_A(T_A + D))$  rounds.*

PROOF. We can prove, in a similar way as the proof of Lemma 10, that execution  $\varrho$  reaches within  $O(T_A + D)$  rounds a configuration in  $\mathcal{C}_{\{2\}}$  where no process is  $A$ -Enabled. If  $v.\bar{x} = v.x$  holds for all  $v \in V$  and  $x \in O_A$ , the network reaches a configuration in  $\mathcal{C}_{\text{fin}}$  by  $L_{12}$  within the next  $O(D)$  rounds. Otherwise, all processes  $v$  perform  $v.\bar{x} \leftarrow v.x$  by  $L_{14}$ , and the network returns to a configuration in  $\mathcal{S}_A$  by  $L_{16}$  within  $O(D)$  rounds. The above execution is repeated until reaching a configuration in  $\mathcal{C}_{\text{fin}}$ , and the number of iterations is bounded by  $L_A$ .  $\square$

**Theorem 1**  *$\text{Loop}(A, E, \mathcal{P})$  is silent and self-stabilizing for  $\mathcal{L}$ . The execution of  $\text{Loop}(A, E, \mathcal{P})$  terminates within  $O(n + T_{\mathcal{P}} + L_A(T_A + D))$  rounds.*

PROOF. Obtained from Lemmas 1, 9, 10, and 11.  $\square$

One may think that a classical self-stabilizing reset algorithm (such as the one described in [8]) can be used to simplify  $\text{Loop}(A, E, \mathcal{P})$  instead of using the five-color waves. However, a self-stabilizing reset algorithm cannot be used directly to implement  $\text{Loop}(A, E, \mathcal{P})$ . A reset algorithm aims to reset or initialize the network to a legal initial configuration (after the illegal configuration is detected) while  $\text{Loop}(A, E, \mathcal{P})$  aims to execute the base algorithm  $A$  repeatedly. Unlike reset algorithms, a reset signal in Loop is not used to initialize the network; a reset signal itself does not trigger an initialization algorithm  $\mathcal{P}$ . Instead, a reset signal is used just to change the color of the root to zero (i.e. to initiate the color waves). Five kinds of color waves described above guarantee that the  $i+1^{\text{th}}$  execution of the base algorithm starts only after the  $i^{\text{th}}$  execution terminates.

## 4. MINIMAL $k$ -GROUPING ALGORITHM

In this section, we give a silent self-stabilizing algorithm for minimal  $k$ -grouping using the loop composition method described in the previous section. In the following, we call a set of processes a *group*. Two distinct groups  $g_1, g_2 \subseteq V$  are said to be *mergeable* if  $D(G(g_1 \cup g_2)) \leq k$ . Two distinct groups  $g_1, g_2 \subseteq V$  are *near* if  $\{v_1, v_2\} \in E$  holds for some  $v_1 \in g_1$  and  $v_2 \in g_2$  and every  $u_1 \in g_1$  and  $u_2 \in g_2$  are within  $k$  hops in  $G$  (not necessarily in  $G(g_1 \cup g_2)$ ). Note that two groups are mergeable only if they are near, but two groups are not necessarily mergeable even if they are near.

## 4.1 Overview

The proposed algorithm  $\text{Loop}(Merge, E, Init)$  consists of two algorithms *Init* and *Merge*. Roughly speaking, *Init* makes an initial partition  $g_1, \dots, g_s$  where  $s \leq 2n/k+1$ , and *Merge* merges two groups as long as two mergeable groups exist. Specifically, in an execution of *Merge*, all processes of each group  $g_i$  first agree on choosing one of  $g_i$ 's near groups, say  $g_j$ , as the target, and next check whether  $g_i$  and  $g_j$  are mergeable. If  $g_i$  and  $g_j$  are mergeable and  $g_j$  also targets  $g_i$ , then they are merged; if they are not mergeable, the *stamp* indicating they are not mergeable is generated on the memories of all the processes of  $g_i$  and  $g_j$ . This stamp removes  $g_j$  from the set of target candidates of  $g_i$  in the following executions of *Merge*. However, this stamp is not permanent; it is removed when  $g_i$  and/or  $g_j$  merges with another group. This removal is needed because the two groups stamped as unmergeable may become mergeable when one of them is merged to another group. After a finite number of executions of *Merge*, the network reaches a configuration in  $\mathcal{C}_{\text{goal}}(\text{Merge}, E)$ , in which all the groups have stamps for all their near groups. As we shall see later, we assign *prior* labels to each group so that prior groups have higher priority of becoming targets than non-prior groups. This strategy is to ensure that the number of executions of *Merge* is bounded by  $O(n/k)$ .

## 4.2 Preliminaries

Actions of *Init* and *Merge* are given in Tables 3 and 6 respectively. By the definition of the composition algorithm, we can use  $v.\text{parent}$  (thus  $\text{Par}(v)$  and  $\text{Chi}(v)$ ) obtained from algorithm BFS in *Init* and *Merge*, and assume that the network remains in  $\mathcal{L}_{\text{BFS}}$  in what follows. Except for copying variables, each process  $v$  has four non-array variables and the following thirteen array variables, as shown in Table 2. Every copying variable  $\bar{x}$  has the same range as the corresponding variable  $x \in O_{\text{Merge}}$ . In Tables 3 and 6, “ $v.x \leftarrow \chi(v)$ ” means “ $v.x \neq \chi(v) \rightarrow v.x \leftarrow \chi(v)$ ”, and “ $v.x[u] \leftarrow \chi(v, u)$ ” means “ $\exists u \in v.\text{domain} : v.x[u] \neq f(\chi(v, u)) \rightarrow v.x[u] \leftarrow f(\chi(v, u))$ ” for all  $u \in v.\text{domain}$ ” where  $f(a) = a$  if  $a$  belongs to the range of  $v.x[u]$ ; otherwise  $f(a) = \perp$  for any  $a$ . We define  $\min \emptyset = \perp$  and  $\min S \cup \perp = \min S$  for any set  $S$ .

We denote  $l_v = v.\text{group}$ ,  $N_v(u) = \{w \in N_v \mid l_w = u\}$ ,  $S_v = N_v(l_v)$ ,  $g_v(u) = \{w \in v.\text{domain} \mid v.\text{groups}[w] = u\}$ , and  $g(u) = \{w \in V \mid w.\text{group} = u\}$ . Note that the value of these notations are based on copying variables *group* and *groups*, and the values of  $l_v$ ,  $N_v(u)$ ,  $S_v$ ,  $g_v(u)$  are computable locally at process  $v$  while we use  $g(u)$  only in explanations and proofs.

We use three macros **Share**, **Min**, and **Distance**. For

**Table 3:** *Init*

<b>[Actions of process <math>v</math>]</b>	
I <sub>1</sub>	$v.\underline{\text{domain}} \leftarrow \text{Domain}(v)$
I <sub>2</sub>	$v.\underline{\text{dist}}[u] \leftarrow \text{Dist}(v, u)$
I <sub>3</sub>	$v.\underline{\text{height}} \leftarrow \text{Height}(v)$
I <sub>4</sub>	$v.\underline{\text{initGroup}} \leftarrow \text{InitGroup}(v)$
I <sub>5</sub>	$v.\underline{\text{group}} \leftarrow \text{InitGroup}(v)$
I <sub>6</sub>	$v.\underline{\text{groups}}[u] \leftarrow \text{Share}(v, u, \underline{\text{groups}}, v.\underline{\text{group}})$
I <sub>7</sub>	$v.\underline{\text{groupD}}[u] \leftarrow \text{Distance}(v, u, \underline{\text{groupD}}, S_v)$
I <sub>8</sub>	$v.\underline{\text{stampON}}[u] \leftarrow \text{false}$
I <sub>9</sub>	$v.\underline{\text{prior}}[u] \leftarrow \text{false}$

**Table 4: Functions used to describe *Init***

$\text{Dist}(v, u) = \text{Distance}(v, u, \text{dist}, \{w \in N_v \mid u \in w.\underline{\text{domain}}\}),$
$\text{Domain}(v) = \{u \mid \text{Dist}(v, u) \neq \perp, \text{Dist}(v, u) \leq k + 1\},$
$\text{Height}(v) = \begin{cases} 0 & \text{if }  \text{Chi}(v)  = 0 \\ \max\{u.\underline{\text{height}} + 1 \bmod \lfloor k/2 + 1 \rfloor \mid u \in \text{Chi}(v)\} & \text{otherwise,} \end{cases}$
$\text{InitGroup}(v) = \begin{cases} v & \text{if }  \text{Par}(v)  = 0 \vee v.\underline{\text{height}} = \lfloor k/2 \rfloor \\ (\underline{v.parent}).\underline{\text{initGroup}} & \text{otherwise.} \end{cases}$

$v, u \in V$ , array variable  $x$ , and the function  $f'$  on the variables of a process, we define  $\text{Share}(v, u, x, f'(v)) = f'(v)$  if  $v = u$ ; otherwise  $\text{Share}(v, u, x, f'(v)) = \min\{w.x \mid w \in N_v, v.\underline{\text{dist}}[u] = w.\underline{\text{dist}}[u] + 1\}$ . The action “ $v.x[u] \leftarrow \text{Share}(v, u, x, f'(v))$ ” lets every process  $v \in V$  stores  $f'(u)$  on  $v.x[u]$  for every  $u \in N_v^{k+1}$ . For example, for every  $v \in V$  and  $u \in N_v^{k+1}$ ,  $v$  eventually has  $v.\underline{\text{groups}}[u] = u.\underline{\text{group}}$  by action M<sub>9</sub> in Table 6. For  $v \in V$ , variable  $x \in ID$ , and predicate  $Q$  on variables of a process, we define  $\text{Min}(v, x, Q(v)) = \min\{v.w \mid Q(v) \text{ holds}\}$ ; otherwise  $\text{Min}(v, x, Q(v)) = w$ , where  $w = \min\{u.x \mid u \in S_v, v.\underline{\text{groupD}}[u.x] = u.\underline{\text{groupD}}[u.x] + 1\}$ . The action “ $v.x \leftarrow \text{Min}(v, x, Q(v))$ ” lets process  $v$  store on  $v.x$  the minimum identifier of the processes satisfying  $Q$  in its group;  $v$  stores  $\perp$  on  $v.x$  if no process of the group satisfies  $Q$ . For example, for every  $v \in V$  and  $u \in N_v^{k+1}$ , action M<sub>1</sub> in Table 6 lets  $v$  store on  $v.\underline{\text{border}}[u]$  the process with the minimum identifier in group  $l_v$  which neighbors to a process in group  $u$ , if such a process exists. For  $u, v \in V$ , array variable  $x$ , and set  $X \subseteq N_v$ , we define  $\text{Distance}(v, u, x, X) = 0$  if  $u = v$ ; otherwise  $\text{Distance}(v, u, x, X) = 1 + \min_{w \in X} w.x[w]$ . By an abuse of notation, we define  $1 + \perp = \perp$  throughout the paper. Hence,  $\text{Distance}(v, u, x, X) = \perp$  if  $v \neq u$  and  $X = \emptyset$ . This macro is useful to compute some kind of distance between  $u$  and  $v$ . For example, for every  $v \in V$  and  $u \in S_v$ ,  $v$  eventually has  $v.\underline{\text{groupD}}[u] = d_{S_v}(u, v)$  by action I<sub>7</sub> in Table 3.

### 4.3 Algorithm *Init*

Algorithm *Init* creates an initial partition based on the BFS-tree, and computes the values of the variables used in *Merge*, such as *domain* and *dist*. The functions used to describe *Init* in Table 3 are defined in Table 4. Algorithm *Init* first computes and stores  $N_v^{k+1}$  on  $v.\underline{\text{domain}}$  and stores  $d(v, u)$  on  $v.\underline{\text{dist}}[u]$  for each  $u \in N_v^{k+1}$  within  $O(k)$  rounds (I<sub>1</sub> and I<sub>2</sub>). Next, *Init* gives an initial partition based on heights of processes in the BFS-tree and stores the group identifier on each  $v.\underline{\text{initGroup}}$  (I<sub>3</sub> and I<sub>4</sub>). After that, *Init* initializes five copying variables *group*, *groups*, *groupD*, *stampON*, and *prior* (I<sub>5</sub>, I<sub>6</sub>, I<sub>7</sub>, I<sub>8</sub>, and I<sub>9</sub>). The number of initial groups is at most  $2n/k + 1$  since every initial group has at least  $k/2$  processes, except for the group including the root of the BFS-tree.

We define error-detecting variable  $E(v)$  in Table 5. (Ig-

nore  $\text{StampOK}(v, u)$  before the middle of Section 4.4.) The following two lemmas hold trivially by the definition of  $\mathcal{E} = \bigvee_{v \in V} E(v)$  (in spite of the definition of  $\text{StampOK}(v, u)$ ).

**Lemma 12** *Every maximal execution of *Init* terminates at a configuration in  $\neg\mathcal{E}$  within  $O(k)$  rounds.*

**Lemma 13** *The number of groups, i.e.  $|\{l_v \mid v \in V\}|$ , is no more than  $2n/k + 1$  at a configuration in  $\neg\mathcal{E}$ .*

### 4.4 Algorithm *Merge*

Assuming the network remains at configurations in  $\neg\mathcal{E}$ , *Merge* merges two near groups if mergeable and/or stamps them if they are not mergeable: notice that the stamps consist of three variables *stamp1*, *stamp2*, and *stampD*. The functions used to describe *Merge* in Table 6 are defined in Table 7.

A group  $g(i)$  is called  $g(j)$ 's *candidate* if  $g(i)$  is near to  $g(j)$  and  $j.\underline{\text{stampON}}(i)$  does not hold, and  $g(i)$  is said to be *prior* if  $i.\underline{\text{prior}}[i]$  holds; otherwise  $g(i)$  is *non-prior*. Note that, if  $g(i)$  is near to  $g(j)$ , all nodes  $v \in g(j)$  agree on their *stampON* and *prior*, and  $v.\underline{\text{prior}}[i] = i.\underline{\text{prior}}[i]$  since we assume that the network stays in  $\neg\mathcal{E}$ .

In actions M<sub>1</sub>, ..., M<sub>7</sub>, each group chooses one of its candidates and checks mergeability with it. Specifically, all processes of each group  $g(i)$  agree to choose the same group  $g(j)$  as their target;  $g(j)$  is the prior candidate that has the minimum  $j$  if such a candidate exists; otherwise  $g(j)$  is the candidate that has the minimum  $j$  (M<sub>1</sub>, M<sub>2</sub>, and M<sub>3</sub>). After actions M<sub>1</sub>, ..., M<sub>3</sub> converge<sup>2</sup>, either *Detector*( $j, i$ ) or *Detector*( $i, j$ ) holds since  $j = \text{Target}(i)$ . In what follows, we assume *Detector*( $j, i$ ) without loss of generality. Then, action M<sub>4</sub> leads to  $v.\underline{\text{merged}}[u] = d_{g(i) \cup g(j)}(v, u)$  for all  $v \in g(j)$  and  $u \in g(i)$  (Lemma 15). Let  $v_{\text{find}} = \min\{v \in g(j) \mid \exists u \in g(i) : d_{g(i) \cup g(j)}(v, u) = k + 1\}$  and  $v_{\text{found}} = \min\{u \in g(i) \mid d_{g(i) \cup g(j)}(u, v_{\text{find}}) = k + 1\}$ . If  $g(i)$  and  $g(j)$  are unmergeable, all processes  $v \in g(j)$  compute  $v_{\text{find}}$  using *merged*, and make the stamp such that  $(v.\underline{\text{stamp1}}[i], v.\underline{\text{stamp2}}[i], v.\underline{\text{stampD}}[i]) = (v_{\text{find}}, \perp, d_{g(i) \cup g(j)}(v, v_{\text{find}}))$ , and all processes  $u \in g(i)$  make the stamp such that  $(u.\underline{\text{stamp1}}[j], u.\underline{\text{stamp2}}[j], u.\underline{\text{stampD}}[j]) =$

<sup>2</sup>We say that actions  $A_1, A_2, \dots, A_s$  converge if none of them is enabled at any process.

**Table 5: Error-detecting variable  $E(v)$**

$E(v) \equiv \neg((v.\text{domain} = \text{Domain}(v)) \wedge (\forall u \in v.\text{domain} : v.\text{dist}[u] = \text{Dist}(v, u))$ $\wedge (v.\text{height} = \text{Height}(v)) \wedge (v.\text{initGroup} = \text{InitGroup}(v))$ $\wedge \text{GrpOK}(v) \wedge \text{GrpsOK}(v) \wedge \text{GrpDistOK}(v)$ $\wedge \forall u \in v.\text{domain} : v.\overline{\text{stampON}}[u] \Rightarrow \text{StampOK}(v, u)$ $\wedge (\forall u \in N_v, \forall w \in v.\text{domain} \cap u.\text{domain} : v.\overline{\text{prior}}[w] = u.\overline{\text{prior}}[w]),$ $\text{GrpOK}(v) \equiv \forall u \in N_v : u.\text{initGroup} = v.\text{initGroup} \Rightarrow u.\overline{\text{group}} = v.\overline{\text{group}},$ $\text{GrpsOK}(v) \equiv \forall u \in v.\text{domain} : v.\overline{\text{groups}}[u] = \text{Share}(v, u, \overline{\text{groups}}, v.\overline{\text{group}}) \wedge l_v \in g_v(l_v),$ $\text{GrpDistOK}(v) \equiv \forall u \in g_v(l_v) : v.\overline{\text{groupD}}[u] = \text{Distance}(v, u, \overline{\text{groupD}}, S_v) \neq k + 1,$ $\text{StampOK}(v, u) \equiv \forall w \in S_v : w.\overline{\text{stampON}}[u] \wedge \forall w \in N_v(u) : w.\overline{\text{stampON}}[l_v]$ $\wedge (s_1 \neq \perp \vee s_2 \neq \perp) \wedge s_D \neq \perp \wedge (s_2 = \perp \Rightarrow \forall w \in N_v(u) : w.\overline{\text{stamp2}}[l_v] \neq \perp)$ $\wedge \forall w \in S_v : (s_1, s_2) = (w.\overline{\text{stamp1}}[u], w.\overline{\text{stamp2}}[u])$ $\wedge (s_2 = v \Rightarrow s_D = k + 1) \wedge \forall i \in \{1, 2\} : (s_i \neq \perp \Rightarrow s_i \in g_v(l_v))$ $\wedge ((s_1, s_2, s_D) \neq (v, \perp, 0) \Rightarrow$ $s_D = 1 + \min(\{w.\overline{\text{stampD}}[u] \mid w \in S_v\} \cup \{w.\overline{\text{stampD}}[l_v] \mid w \in N_v(u)\}))$
where $(s_1, s_2, s_D) = (v.\overline{\text{stamp2}}[u], v.\overline{\text{stamp2}}[u], v.\overline{\text{stampD}}[u]).$

$(\perp, v_{\text{found}}, d_{g(i) \cup g(j)}(u, v_{\text{find}}))$  ( $M_5, \dots, M_7$ ). If  $g(i)$  and  $g(j)$  are mergeable, then  $v_{\text{find}} = \perp$ , thus all processes  $v \in g(j)$  and  $u \in g(i)$  store  $\perp$  on  $v.\overline{\text{stampD}}[i]$  and  $u.\overline{\text{stampD}}[j]$  ( $M_5$  and  $M_6$ ). Old stamps between two groups  $g(i')$  and  $g(j')$  remain when  $\text{Target}(i') \neq j'$  and  $\text{Target}(j') \neq i'$  hold ( $M_5, \dots, M_7$ ).

After actions  $M_1, \dots, M_7$  converge, two groups  $g(i)$  and  $g(j)$  merge by  $M_8$  (i.e.  $v.\text{group} = \min\{i, j\}$  for all  $v \in g(i) \cup g(j)$ ) if and only if the two groups are mergeable and they target each other, which can be easily determined from the values of **target** and **stampD**. When a group is not merged, all processes  $v$  of the group remain at the same group (i.e. we have  $v.\text{group} = v.\overline{\text{group}}$ ). Thereafter, **groups** and **groupD** are updated by  $M_9$  and  $M_{10}$ . The variable **stampON** represents the validity of the stamp. If a group merges with its target, the stamps regarding the group should be removed. Hence,  $v.\overline{\text{stampON}}[u]$  is set to false not only when  $v.\overline{\text{stampD}}[u] = \perp$  but also when  $v$  and/or  $u$  merges with some group; otherwise  $v.\overline{\text{stampON}}[u]$  is set to true ( $M_{11}$  and  $M_{12}$ ). Variable **prior** is updated by the following simple policy ( $M_{13}$ ): a non-prior group becomes a prior group when it merges with its target, and a prior group becomes a non-prior group when its  $\overline{\text{stampON}}[i]$  holds for every near group  $g(i)$ . Thanks to this policy, the number of iterations of **Merge** is bounded by  $O(n/k)$  (Lemma 22).

Now, let us see the definition of  $\text{StampOK}(v, u)$  in Table 5. Suppose that an execution of **Merge** terminates at some configuration  $\gamma$ . Then, it is easily shown that  $v.\overline{\text{stampON}}[u] \Rightarrow \text{StampOK}(v, u)$  holds for all  $v \in V$  and  $u \in v.\text{domain}$  in  $\gamma^{\text{copy}}$ . (See Lemma 20.)

**Lemma 14** Consider two groups  $g(i)$  and  $g(j)$  are near in a configuration  $\gamma \in \neg\mathcal{E}$ . Then,  $g(i)$  and  $g(j)$  are not mergeable if  $v.\overline{\text{stampON}}[j]$  holds for some  $v \in g(i)$ .

PROOF. Since  $\gamma \in \neg\mathcal{E}$  holds, (i) all processes  $w_1 \in g(i)$  have a common non-null  $v_{\text{find}} = w_1.\overline{\text{stamp1}}[j] \in g(i)$  and all processes  $w_2 \in g(j)$  have a common non-null  $v_{\text{found}} = w_2.\overline{\text{stamp2}}[i] \in g(j)$ , or (ii) all processes  $w_1 \in g(i)$  have a common non-null  $v_{\text{found}} = w_1.\overline{\text{stamp2}}[j] \in g(i)$  and all processes  $w_2 \in g(j)$  have a common non-null  $v_{\text{find}} = w_2.\overline{\text{stamp1}}[i] \in g(j)$ . In both cases,  $\neg\mathcal{E}$  guarantees  $d_{g(i) \cup g(j)}(v_{\text{find}}, v_{\text{found}}) = k + 1$ , which means  $D(G(g(i) \cup g(j))) \geq k + 1$ .  $\square$

**Lemma 15** Let  $\gamma$  be a configuration in  $\neg\mathcal{E}$  where  $M_1, \dots, M_4$  converges and  $j = \text{Target}(i)$ . In configuration  $\gamma$ , we have  $v.\overline{\text{mergeD}}[u] = d_{g(i) \cup g(j)}(v, u)$  for all  $v \in g(j)$  and  $u \in g(i)$ .

PROOF. We let  $d(v) = d_{g(i) \cup g(j)}(v, u)$  for any fixed  $u \in g(i)$ . Clearly,  $u.\overline{\text{mergeD}}[u] = d(u) = 0$  holds in  $\gamma$ . Let  $w$  be a process of  $g(j) \cup g(i) \setminus \{u\}$ . We have  $w.\overline{\text{mergeD}}[u] \geq d(w)$  since  $w.\overline{\text{mergeD}}[u] = 1 + \min\{w'.\overline{\text{mergeD}}[u] \mid w' \in N_w(i) \cup N_w(j)\}$  and  $w''.\overline{\text{mergeD}}[u] = 0$  only if  $w'' \neq u$ . On the other hand, since there exists a path  $w, w_1, w_2, \dots, u$  of length  $d(w)$  in  $G(g(i) \cup g(u))$ , we have  $w.\overline{\text{mergeD}}[u] \leq 1 + w_1.\overline{\text{mergeD}}[u] \leq 2 + w_2.\overline{\text{mergeD}}[u] \leq \dots \leq d(w) + u.\overline{\text{mergeD}}[u] = d(w)$ .  $\square$

Let  $x_i$  be the variable updated by action  $M_i$ , and  $\mathcal{G}_i(v)$  be the guard for action  $M_i$ . (e.g. variable  $x_8$  denotes **group** and guard  $\mathcal{G}_{13}(v)$  denotes  $\exists u \in v.\text{domain} : v.\overline{\text{prior}}[u] \neq \text{Share}(v, u, \overline{\text{prior}}, \text{Prior}(v))$ .)

**Lemma 16 (Shiftable Convergence)** Every maximal execution  $\varrho = \gamma_0, \gamma_1, \dots$  of **Merge** where  $\gamma_0 \in \neg\mathcal{E}$  terminates at configuration  $\gamma$  such that  $\gamma^{\text{copy}} \in \neg\mathcal{E}$ , within  $O(k)$  rounds.

PROOF. It is guaranteed by Lemma 17, as shown below, that  $\varrho$  terminates within  $O(k)$  rounds. Hence, it suffices to show  $\gamma^{\text{copy}} \in \neg\mathcal{E}$  where  $\gamma$  is the configuration at which  $\varrho$  terminates. Recall that  $\mathcal{E} \equiv \bigvee_{v \in V} E(v)$ , and  $E(v)$  is defined as follows: Since  $\gamma_0 \in \neg\mathcal{E}$  and execution  $\varrho$  never updates variables **domain**, **dist**, **height**, and **initGroup**, configuration  $\gamma^{\text{copy}}$  satisfies  $(v.\text{domain} = \text{Domain}(v)) \wedge (\forall u \in v.\text{domain} : v.\text{dist}[u] = \text{Dist}(v, u)) \wedge (v.\text{height} = \text{Height}(v)) \wedge (v.\text{initGroup} = \text{InitGroup}(v))$  for any  $v \in V$ . Since  $\varrho$  never assigns different groups to distinct processes in the same group at  $\gamma_0$ ,  $\gamma^{\text{copy}}$  satisfies  $\text{GrpOK}(v)$  for any  $v \in V$ . By Lemmas 18, 19, and 20 shown below, in configuration  $\gamma^{\text{copy}}$ , we have  $\text{GrpsOK}(v)$ ,  $\text{GrpDistOK}(v)$ , and  $\forall u \in v.\text{domain} : v.\overline{\text{stampON}}[u] \Rightarrow \text{StampOK}(v, u)$  for any  $v \in V$ . We also have  $\forall u \in N_v, \forall w \in v.\text{domain} \cap u.\text{domain} : v.\overline{\text{prior}}[w] = u.\overline{\text{prior}}[w]$  for any  $v \in V$  in  $\gamma^{\text{copy}}$ , since  $\gamma$  satisfies  $v.\overline{\text{prior}}[u] = \text{Share}(v, u, \overline{\text{prior}}, \text{Prior}(v))$  for all  $v \in V$  and  $u \in v.\text{domain}$ .  $\square$

**Lemma 17** Every maximal execution  $\varrho = \gamma_0, \gamma_1, \dots$  of **Merge** where  $\gamma_0 \in \neg\mathcal{E}$  terminates within  $O(k)$  rounds.

**Table 6: Merge**

[Actions of process $v$ ]		
M <sub>1</sub>	$v.\text{border}[u] \leftarrow \text{Min}(v, \text{border}[u], N_v(u) \neq \emptyset)$	
M <sub>2</sub>	$v.\text{far}[u] \leftarrow \text{Min}(v, \text{far}[u], \exists w \in g_v(u) : v.\text{dist}[w] = k + 1)$	
M <sub>3</sub>	$v.\text{target}[u] \leftarrow \text{Share}(v, u, \text{target}, \text{Target}(v))$	
M <sub>4</sub>	$v.\text{mergeD}[u] \leftarrow \text{MergeDist}(v, u)$	
M <sub>5</sub>	$v.\text{stamp1}[u] \leftarrow \text{Stamp}_1(v, u)$	
M <sub>6</sub>	$v.\text{stampD}[u] \leftarrow \text{Stamp}_D(v, u)$	
M <sub>7</sub>	$v.\text{stamp2}[u] \leftarrow \text{Min}(v, \text{stamp2}[u], \text{Stamp}_D(v, u) = k + 1)$	
M <sub>8</sub>	$v.\text{group} \leftarrow \text{Group}(v)$	
M <sub>9</sub>	$v.\text{groups}[u] \leftarrow \text{Share}(v, u, \text{groups}, v.\text{group})$	
M <sub>10</sub>	$v.\text{groupD}[u] \leftarrow \text{Distance}(v, u, \text{groupD}, \{w \in N_v \mid w.\text{group} = v.\text{group}\})$	
M <sub>11</sub>	$v.\text{merging}[u] \leftarrow \text{Share}(v, u, \text{merging}, \text{Merging}(v))$	
M <sub>12</sub>	$v.\text{stampON}[u] \leftarrow v.\text{stampD}[u] \neq \perp \wedge \neg \text{Merging}(v) \wedge \neg v.\text{merging}[u]$	
M <sub>13</sub>	$v.\text{prior}[u] \leftarrow \text{Share}(v, u, \text{prior}, \text{Prior}(v))$	

**Table 7: Functions used to describe Merge**

$Cand(v) = \{u \in v.\text{domain} \setminus \{l_v\} : v.\text{border}[u] \neq \perp \wedge v.\text{far}[u] = \perp \wedge \neg v.\text{stampON}[u]\}$
$Target(v) = \begin{cases} \min\{u \in Cand(v) \mid v.\text{prior}[u]\} & \text{if } \exists u \in Cand(v) : v.\text{prior}[u] \\ \min Cand(v) & \text{otherwise} \end{cases}$
$MergeDist(v, u) = \begin{cases} \text{Distance}(v, u, \text{mergeD}, S_v \cup N_v(Target(v))) & \text{if } u \in g_v(l_v) \\ \text{Distance}(v, u, \text{mergeD}, S_v \cup N_v(v.\text{groups}[u])) & \text{otherwise} \end{cases}$
$Detector(v, u) \equiv l_v = v.\text{target}[u] \wedge (l_v \leq u \vee Target(v) \neq u)$
$Stamp_1(v, u) = \begin{cases} \text{Min}(v, \text{stamp1}[u], \exists w \in g_v(u) : v.\text{mergeD}[w] = k + 1) & \text{if } Detector(v, u) \\ v.\text{stamp1}[u] & \text{if } v.\text{stampON}[u] \\ \perp & \text{otherwise} \end{cases}$
$Stamp_D(v, u) = \begin{cases} 0 & \text{if } v = Stamp_1(v, u) \\ 1 + \min \left( \{w.\text{stampD}[u] \mid w \in S_v\} \cup \{w.\text{stampD}[l_v] \mid w \in N_v(u)\} \right) & \text{otherwise} \end{cases}$
$Merging(v) \equiv l_v = v.\text{target}[Target(v)] \wedge v.\text{stampD}[Target(v)] = \perp$
$Group(v) = \begin{cases} \min\{g_v(v), Target(v)\} & \text{if } Merging(v) \\ g_v(v) & \text{otherwise} \end{cases}$
$Saturated(v) \equiv \forall u \in v.\text{domain} : (v.\text{border}[u] \neq \perp \wedge v.\text{far}[u] = \perp) \Rightarrow v.\text{stampON}[u]$
$Prior(v) \equiv Merging(v) \vee (v.\text{prior}[v] \wedge \neg Saturated(v))$

PROOF. Guard  $\mathcal{G}_i(v)$  is independent from the value of  $u.x_j$  for any  $u \in N_v^1$  and  $j > i$ . Hence, in an execution of *Merge*,  $v.x_j$  is never updated after actions  $M_0, \dots, M_j$  converged (i.e.  $\forall u \in V, i \in [0, j] : G_i(u) = \text{false}$ ). Therefore, by the description of *Merge*, any action  $M_j$  converges within  $O(k)$  rounds after actions  $M_0, \dots, M_{j-1}$ . This proves the lemma.  $\square$

**Lemma 18** *If a maximal execution  $\varrho = \gamma_0, \gamma_1, \dots$  of Merge where  $\gamma_0 \in \neg \mathcal{E}$  terminates at configuration  $\gamma$ , then  $\gamma^{\text{copy}}$  satisfies  $\text{GrpsOK}(v)$  for all  $v \in V$ .*

PROOF. Recall the definition of  $\text{GrpsOK}(v)$  in Table 5. By that definition, it suffices to show that, in configuration  $\gamma$ , we have  $v.\text{groups}[u] = \text{Share}(v, u, \text{groups}, v.\text{group})$  for all  $u \in v.\text{domain}$  and  $v.\text{groups}[v.\text{group}] = v.\text{group}$ . The former condition holds since  $\mathcal{G}_9$  holds in  $\gamma$ . The latter condition holds if  $g(l_v)$  is not merged in execution  $\varrho$ , since  $v.\text{group} = l.\text{group} = l$  holds in  $\gamma$ . Even if  $g(l_v)$  merges with  $g(u)$  in  $\varrho$ , the latter condition holds since  $\mathcal{G}_8$  guarantees  $v.\text{group} = \min\{l_v, u\}.\text{group} = \min\{l_v, u\}$  in  $\gamma$ .  $\square$

**Lemma 19** *If a maximal execution  $\varrho = \gamma_0, \gamma_1, \dots$  of Merge where  $\gamma_0 \in \neg \mathcal{E}$  terminates at configuration  $\gamma$ , then  $\gamma^{\text{copy}}$  satisfies  $\text{GrpDistOK}(v)$  for all  $v \in V$ .*

PROOF. Recall the definition of  $\text{GrpDistOK}(v)$  in Table 5. By that definition, it suffices to show that, in configuration  $\gamma$ , we have  $v.\text{groupD}[u] = \text{Distance}(v, u, \{w.\text{groupD}[u] \mid w \in N_v, w.\text{group} = v.\text{group}\})$  and  $v.\text{groupD}[u] \neq k + 1$  for all  $u \in \{w \in N_v \mid w.\text{group} = v.\text{group}\}$ . The former condition holds since  $\mathcal{G}_{10}$  holds in  $\gamma$ . The latter condition holds if  $g(l_v)$  is not merged in execution  $\varrho$ , since  $v.\text{groupD}[u] = d_{g(l_v)}(v, u) \in [0, k]$  holds in  $\gamma$ . If  $g(l_v)$  merges with some group  $g(w)$  in  $\varrho$ , then  $d_{g(l_v) \cup g(w)}(v, u) \in [0, k]$  holds; otherwise  $g(l_v)$  cannot be merged with  $g(w)$ . Hence, even if  $g(l_v)$  merges with  $g(w)$  in  $\varrho$ , the latter condition holds since  $\mathcal{G}_{10}$  guarantees  $v.\text{groupD}[u] = d_{g(l_v) \cup g(w)}(v, u) \in [0, k]$  in  $\gamma$ .  $\square$

**Lemma 20** *If a maximal execution  $\varrho = \gamma_0, \gamma_1, \dots$  of Merge where  $\gamma_0 \in \neg \mathcal{E}$  terminates at configuration  $\gamma$ , then  $\gamma^{\text{copy}}$  satisfies  $v.\text{stampON}[u] \Rightarrow \text{StampOK}(v, u)$  for all  $v \in V$  and*

$u \in v.\text{domain}$ .

PROOF. Recall the definition of  $\text{StampOK}(v)$  in Table 5. Assuming that  $v.\overline{\text{stampON}}[u]$  holds in  $\gamma^{\text{copy}}$ , we will prove that  $\text{StampOK}(v, u)$  holds in  $\gamma^{\text{copy}}$ . Only in this proof, we denote  $w_1.\text{stamp1}[u]$ ,  $w_1.\text{stamp2}[u]$ , and  $w_1.\text{stampD}[u]$  by just  $w_1.\text{stamp1}$ ,  $w_1.\text{stamp2}$ , and  $w_1.\text{stampD}$  for all  $w_1 \in g(l_v)$ , and we denote  $w_2.\text{stamp1}[l_v]$ ,  $w_2.\text{stamp2}[l_v]$ , and  $w_2.\text{stampD}[l_v]$  by just  $w_2.\text{stamp1}$ ,  $w_2.\text{stamp2}$ , and  $w_2.\text{stampD}$  for all  $w_2 \in g(u)$ . Since  $v.\overline{\text{stampON}}[u]$  holds in  $\gamma$ , either  $g(l_v)$  or  $g(u)$  never merges in execution  $\varrho$ , and there exists  $v_{\text{find}} \in g(l_v) \cup g(u)$  such that  $w.\text{stampD} = d_{g(l_v) \cup g(u)}(w, v_{\text{find}})$  holds for all  $w \in g(l_v) \cup g(u)$ . Let  $v_{\text{found}} = \min\{w \in g(l_v) \cup g(u) \mid d_{g(l_v) \cup g(u)}(w, v_{\text{find}}) = k + 1\}$ . (Note that  $v_{\text{found}} \neq \perp$ .) In  $\gamma$ , we have  $(w.\text{stamp1}, w.\text{stamp2}) = (v_{\text{find}}, \perp)$  for all  $w \in g(v_{\text{found}}.\overline{\text{group}})$ , and  $(w.\text{stamp1}, w.\text{stamp2}) = (\perp, v_{\text{found}})$  for all  $w \in g(v_{\text{found}}.\overline{\text{group}})$ . All of these prove that  $\text{StampOK}(v, u)$  holds.  $\square$

In the rest of this section, we denote  $\mathcal{C}_{\text{goal}}(\text{Merge}, E)$  simply by  $\mathcal{C}_{\text{goal}}$ .

**Lemma 21 (Correctness)** *Every  $\gamma \in \mathcal{C}_{\text{goal}}$  satisfies  $\mathcal{L}_k$ .*

PROOF. Let  $V(i) = \{v \in V \mid v.\text{group} = i\}$ . In  $\gamma \in \mathcal{C}_{\text{goal}}$ , we have (i)  $D(g(i)) \leq k$  for all  $i \in V$  s.t.  $g(i) \neq \emptyset$ , (ii)  $\text{Cand}(v) = \emptyset$  for all  $v \in V$ , and (iii)  $g(i) = V(i)$  for all  $i \in V$ . By Lemma 14 and (ii), every  $g(i)$  is not mergeable with any other group in  $\gamma$ . Hence, by (i) and (iii),  $\{V(i) \neq \emptyset \mid i \in V\}$  is a minimal partition of  $G$  with diameter-bound  $k$  in  $\gamma$ .  $\square$

**Lemma 22 (Loop Convergence)** *Let  $\varrho_0, \varrho_1 \dots$  be an infinite sequence of maximal executions of  $\text{Merge}$  where  $\varrho_i = \gamma_{i,0}, \gamma_{i,1}, \dots, \gamma_{i,s_i}$ ,  $\gamma_{0,0} \in \neg\mathcal{E}$ , and  $\gamma_{i+1,0} = \gamma_{i,s_i}^{\text{copy}}$  for  $i \geq 0$ . Then,  $\gamma_{j,s_j} \in \mathcal{C}_{\text{goal}}$  holds for some  $j = O(n/k)$ .*

**Proof Sketch.** We say that a group  $g(v)$  is black if  $g(v)$  is non-prior and some non-prior group  $g(u)$  exists such that  $u \in \text{Cand}(v)$ ; otherwise  $g(v)$  is white. Note that a black group may become white, but no white group becomes black. We denote the number of groups (*i.e.*  $|\{l_v \mid v \in V\}|$ ), the number of black groups, and the number of prior groups in configuration  $\gamma \in \neg\mathcal{E}$  by  $\#_g(\gamma)$ ,  $\#_b(\gamma)$ , and  $\#_p(\gamma)$  respectively. We define  $\#(\gamma) = \#_g(\gamma) + \#_b(\gamma) + \#_p(\gamma)$ . It is easily shown that  $\#(\gamma_{i,0}) > \#(\gamma_{i+1,0})$  unless  $\gamma_i \in \mathcal{C}_{\text{goal}}$ . By Lemma 13,  $\#(\gamma) \leq 3\#_g(\gamma) = O(n/k)$  holds. Hence, we have  $\gamma_{j,s_j} \in \mathcal{C}_{\text{goal}}$  for some  $j = O(n/k)$ .  $\square$

**Theorem 2** *Algorithm Loop( $\text{Merge}, E, \text{Init}$ ) is silent self-stabilizing for  $\mathcal{L}_k$ . Every maximal execution of the algorithm terminates within  $O(nD/k)$  rounds. The number of groups it produces, *i.e.*  $|\{v.\text{group} \mid v \in V\}|$ , is at most  $2n/k + 1$ . The space complexity per process of Loop( $\text{Merge}, E, \text{Init}$ ) is  $O((n + n_{\text{false}})\log n)$  where  $n_{\text{false}}$  is the number of identifiers stored in  $v.\text{domain}$  for some  $v \in V$  in an initial configuration, and which do not match the identifier of any  $u \in V$ .*

PROOF. The first three arguments are proven by Lemmas 1, 12, 13, 16, 21, and 22. The last argument about the space complexity is trivial.  $\square$

## 5. CONCLUSION

We have given a silent self-stabilizing algorithm for the minimal  $k$ -grouping problem. Given a network  $G$  and a

diameter-bound  $k$ , it guarantees that, regardless of an initial configuration, the network reaches a configuration where the diameter of every group is no more than  $k$  and no two groups can be merged without violating the diameter-bound. Its time complexity is  $O(nD/k)$  and its space complexity per process is  $O((n + n_{\text{false}})\log n)$ . The number of groups it produces is at most  $2n/k + 1$ . A novel composition technique called *loop composition* is also given and used in our algorithm.

## 6. REFERENCES

- [1] A. D. Amis, R. Prakash, T. HP. Vuong, and D. T. Huynh. Max-min d-cluster formation in wireless ad hoc networks. In *Proc. of INFOCOM 2000*, volume 1, pages 32–41, 2000.
- [2] A. K. Datta, S. Gurumurthy, F. Petit, and V. Villain. Self-stabilizing network orientation algorithms in arbitrary rooted networks. In *Proc. of ICDCS 2000*, pages 576–583, 2000.
- [3] A. K. Datta, L. L. Larmore, S. Devismes, K. Heurtefeux, and Y. Rivierre. Competitive self-stabilizing k-clustering. In *Proc. of ICDCS 2012*, pages 476–485, 2012.
- [4] A. K. Datta, L. L. Larmore, S. Devismes, K. Heurtefeux, and Y. Rivierre. Self-stabilizing small k-dominating sets. *International Journal of Networking and Computing*, 3(1):116–136, 2013.
- [5] A. K Datta, L. L Larmore, and P.I. Vemula. An  $O(n)$ -time self-stabilizing leader election algorithm. *Journal of Parallel and Distributed Computing*, 71(11):1532–1544, 2011.
- [6] J. S. Deogun, D. Kratsch, and G. Steiner. An approximation algorithm for clustering graphs with dominating diametral path. *IPL*, 61(3):121–127, 1997.
- [7] EW Dijkstra. Self stabilizing systems in spite of distributed control. *Communications of the ACM*, 17:643–644, 1974.
- [8] S. Dolev. *Self-stabilization*. MIT press, 2000.
- [9] S. Dolev and T. Herman. Parallel composition of stabilizing algorithms. In *Proc. of WSS 1999*, pages 25–32, 1999.
- [10] B. Ducourthial, S. Khalfallah, and F. Petit. Best-effort group service in dynamic networks. In *Proc. of SPAA 2010*, pages 233–242, 2010.
- [11] Y. Fernandes and D. Malkhi. K-clustering in wireless ad hoc networks. In *Proceedings of the second ACM international workshop on Principles of mobile computing*, pages 31–37, 2002.
- [12] T. R. Herman. *Adaptivity through distributed convergence*. PhD thesis, University of Texas at Austin, 1992.
- [13] Y. Yamauchi, S. Kamei, F. Ooshita, Y. Katayama, H. Kakugawa, and T. Masuzawa. Hierarchical composition of self-stabilizing protocols preserving the fault-containment property. *IEICE transactions on information and systems*, 92(3):451–459, 2009.
- [14] Y. Yamauchi, S. Kamei, F. Ooshita, Y. Katayama, H. Kakugawa, and T. Masuzawa. Timer-based composition of fault-containing self-stabilizing protocols. *Information Sciences*, 180(10):1802–1816, 2010.

# Local Checkability in Dynamic Networks

Klaus-Tycho Foerster<sup>\*</sup>  
Microsoft Research, USA  
foklaus@ethz.ch

Oliver Richter  
ETH Zurich, Switzerland  
richtero@ethz.ch

Jochen Seidel  
ETH Zurich, Switzerland  
seidelj@ethz.ch

Roger Wattenhofer  
ETH Zurich, Switzerland  
wattenhofer@ethz.ch

## ABSTRACT

In this work we study local checkability of network properties considering inconsistency throughout the verification process. We use disappearing and appearing edges to model inconsistency and prover-verifier-pairs (PVPs) for verification. We say that a network property  $\mathcal{N}$  is *locally checkable under inconsistency* if there exists a PVP for a specified inconsistency. In such a PVP, a prover  $\mathcal{P}$  assigns a label to each vertex of an initial graph  $G_i$ . A distributed time constant verifier  $\mathcal{V}$  computes YES on every vertex of the altered graph  $G_a$ , if  $G_a$  has the property  $\mathcal{N}$  and was labeled by  $\mathcal{P}$ , and NO on at least one vertex of  $G_a$  if  $G_a$  does not fulfill  $\mathcal{N}$ .

For  $s$ - $t$ -reachability, we present an optimal 2-bit-label PVP considering one disappearing edge and an upper bound of  $\mathcal{O}(\log n)$  bits as label size for one appearing edge. For cyclicity, we prove that no general PVP can be found considering disappearing edges, as well as an upper bound of  $\mathcal{O}(n^2 \log n)$  bits as the label size for one appearing edge. Furthermore, we show that no PVP can be found for  $s$ - $t$ -reachability nor general cyclicity that can consider multiple inconsistencies.

## CCS Concepts

- Networks → Network properties; • Computer systems organization → Dependable and fault-tolerant systems and networks; • Theory of computation → Models of computation; • Computing methodologies → Distributed computing methodologies;

## Keywords

Local Checking, s-t Reachability, Cyclicity, Dynamic Graphs

## 1. INTRODUCTION

A network administrator must know whether the network is correct [20]. E.g., is destination  $t$  reachable from source  $s$ , or may the current forwarding rules send packets in a cycle?

\*Most of the work was done while being at ETH Zurich.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ICDCN '17, January 04-07, 2017, Hyderabad, India

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007779>

Edge Inconsistencies	$s$ - $t$ reachability	Cyclicity (max. 1 cycle)	Cyclicity (general)
One Disappearing	2	2	NLC
One Appearing	$\mathcal{O}(\log n)$	$\mathcal{O}(n^2 \log n)^+$	$\mathcal{O}(n^2 \log n)^+$
Multiple Disappearing	NLC	2	NLC
Multiple Appearing	NLC	NLC	NLC
Combinations	NLC	NLC	NLC

Table 1: Main findings of this work summarized in a table. The table entries show the proof label size (in bits) of a prover-verifier algorithm solving the problem or NLC if the problem is Not Locally Checkable, i.e., not solvable in the model we introduce. Two entries carry the addition “+”, which means that these algorithms are only feasible if the graph vertices carry unique identification tags (IDs). Reachability is covered in Section 3, Cyclicity in Section 4.

Traditionally, such network properties are checked by sending probing packets into the network. It is potentially much less costly to check these *global* network properties by inexpensive *local* verification, e.g. [8, 19]: each node just compares its state with the state of its neighbors, and raises an alarm if it senses that something is not right. Such local checkability is studied and well understood.

In real networks, however, links appear and disappear regularly. Unfortunately, previous work on local checkability does not support such dynamic situations. In this work we explore the possibilities and limitations of local checkability of global network properties in *dynamic* networks. We study disappearing and/or appearing edges. E.g., is there a way to locally determine whether destination  $t$  is still reachable from source  $s$  even after an arbitrary link went down?

We check a property through a distributed nondeterministic algorithm modeled as *prover-verifier-pair* (PVP) as introduced in [8]: A prover  $\mathcal{P}$  assigns specific labels to all vertices (the *proof*) if the graph has (or might get through inconsistencies) the property in question. From then on, a distributed deterministic verifier  $\mathcal{V}$  can verify the property by taking as input on each vertex  $v$  only the label of  $v$  and its immediate neighbors' labels.

The results we found for the problems addressed are summarized in Table 1. Besides these results the contributions of this work are a prover-verifier-pair for checking  $s$ - $t$  reachability if one edge in the graph might disappear or appear between labeling and verification; and the fact that no such pair can be found for checking cyclicity in a graph where one edge might disappear without losing the generality that a graph might have multiple cycles.

Due to the space restrictions for this extended abstract, some proofs are deferred to the full version of this article.

## 2. DEFINITIONS AND MODEL

Networks are modeled as graphs  $G = (V(G), E(G))$  in this article, where  $V(G)$  denotes the set of vertices and  $E(G)$  the set of edges. If the graph  $G$  is clear from the context, a shorter notation  $V = V(G)$  and  $E = E(G)$  is used. This work focuses on *undirected graphs*, i.e., if  $u, v \in V(G)$  then  $e = (u, v) \in E(G)$  is equivalent to  $e' = (v, u)$  or in short  $e = e' = \{u, v\}$ . For any vertex  $v \in V$  we denote the number of edges adjacent to  $v$  by  $\deg(v)$ , called *degree of  $v$* .

The function  $l : V \rightarrow \{0, 1\}^*$  shall denote a (*vertex labeling*) for  $G$  that assigns a finite bit-string as *label* to each vertex in  $V$ . Given a graph  $G$  with labeling  $l$ , let  $M(v) = \{l(u_1), \dots, l(u_{\deg(v)})\}$  denote the *set of messages* vertex  $v$  receives in one communication round, where  $u_1, \dots, u_{\deg(v)} \in V$  are the immediate neighbors of  $v$ . Note that the set  $M(v)$  is unordered and neither vertex identifiers nor port numbers are attached to it. The function  $o_{M(v)} : \{0, 1\}^* \rightarrow \mathbb{N}_0$  shall denote the number of occurrences a given label has in the set of messages  $M(v)$ . If  $M(v)$  is clear from the context we will use the short form  $o(\cdot)$ . Furthermore, in this article logarithms are rounded up to integer value and use base 2.

To model inconsistency,  $G_i$  shall denote the *initial graph*, the graph in question before any inconsistencies are considered, and  $G_a$  shall denote the *altered graph*, the graph after all inconsistencies were considered. Since we only alter graph edges to model inconsistency  $V(G_i) = V(G_a) = V$ . Note that even though the set of vertices  $V$  stays the same, the degree  $\deg(v)$  and set of messages  $M(v)$  for a given  $v \in V$  might change. However, we shall evaluate  $\deg(v)$  and  $M(v)$  solely on the altered graph  $G_a$ .

### 2.1 Inconsistencies

Inconsistency is modeled through *disappearing edges* and *appearing edges*. For  $u, v \in V$ , a *disappearing edge*  $d = \{u, v\} \in E(G_i)$  is an edge that is part of the initial graph  $G_i$  but not of the altered graph, i.e.,  $d \notin E(G_a)$ . For  $x, y \in V$  an *appearing edge*  $a = \{x, y\} \in E(G_a)$  is an edge that is part of the altered graph  $G_a$  but was not in the initial graph, i.e.,  $a \notin E(G_i)$ . Thus,  $E(G_a) = E(G_i) \setminus D \cup A$ , where  $D$  denotes the set of all disappearing edges and  $A$  denotes the set of all appearing edges.

In our model, any vertex  $v \in V(G_a)$  gets an additional 3 bits of information  $R(v) = \{r_1, r_2, r_3\}$  to report inconsistency:  $r_1$  is set if and only if  $v$  has an adjacent disappearing edge,  $r_2$  is set if and only if  $v$  has an adjacent appearing edge, and  $r_3$  is set if and only if there has been an appearing edge, i.e., if  $A \neq \emptyset$ . We will refer to these bits as *first*, *second*, or *third inconsistency reporting bit*, respectively.

### 2.2 Prover-Verifier-Pair

A *network* or *graph property* is defined as a set  $Y$  containing all graphs that posses a common attribute, also referred to as *YES-instances*. Any graph  $G \notin Y$  is referred to as *No-instance*. We verify graph properties in two steps, a *proof step* and a *verification step*.

In the *proof step*, a *prover*  $\mathcal{P}$  gets as input an initial graph  $G_i$  and computes for every vertex  $v \in V(G_i)$  a finite label  $l(v)$ . We refer to this labeling  $l$  as the *proof*.

For the *verification step* let  $G_a$  be any graph and let  $l$  be any labeling for  $G_a$ . A *verifier*  $\mathcal{V}$  is a distributed algorithm that gets as input on every vertex  $v$ : the vertex label  $l(v)$ , the set of messages  $M(v)$  containing the labels of neighboring vertices, and the three inconsistency reporting bits; and

computes as output for each vertex either YES or NO.

A *prover-verifier-pair*  $(\mathcal{P}, \mathcal{V})$  or PVP in short is said to be *correct* for  $Y$  if the following two points hold: First, if  $G_a \in Y$  and  $l$  was obtained from  $\mathcal{P}$ , then  $\mathcal{V}$  computes YES as output for all vertices, and second, if  $G_a \notin Y$ , then  $\mathcal{V}$  computes NO as output for at least one vertex, regardless of the labeling.

A network property  $Y$  is *not locally checkable* (or NLC in short) if there exists no PVP that is correct for  $Y$ . To measure the quality of a PVP we specify the *proof size*  $f(n)$  as the maximal number of bits any vertex of any YES-instance with  $n$  vertices might get assigned by the prover. For a given network property  $Y$ , the smallest proof size for which there exists a correct PVP for  $Y$  defines the *proof size for  $Y$* .

## 3. REACHABILITY

One of the fundamental questions in network theory is the question of *s-t-reachability*: Can a given source vertex  $s \in V$  in the network reach a destination vertex  $t \in V$  in the network? This question is the basis for all data transfer throughout the network since two devices (network vertices) can only communicate if they can reach each other.

More formally:  $s \in V$  can *reach*  $t \in V$  if and only if there exists a path from  $s$  to  $t$  in the underlying network graph.

For two vertices  $u, v \in V$  for which there exists a path from  $u$  to  $v$  we will use the two expressions “ $u$  can *reach*  $v$ ” and “ $u$  is *connected to*  $v$ ” interchangeably.

To make meaningful statements about reachability or connectivity we focus in this section on finite graphs with at least two vertices from which one is carrying the unique label  $s$  and another is carrying the unique label  $t$ , cf. [14].

### 3.1 One Disappearing Edge

First, we take a look at what can happen if at most one edge  $d$  disappears, i.e.,  $E(G_a) = E(G_i) \cup E(G_i) \setminus \{d\}$ , where  $d$  can be any edge of the initial graph.

Here we focus on the case that *s-t*-reachability is given in the initial graph since this is the case where the property might get altered. Therefore let *s-t-REACHABILITY* denote the set of all undirected graphs with a path from  $s$  to  $t$ .

In this subsection we first explain the proof idea and then prove the following theorem:

**THEOREM 1.** *The proof size for s-t-REACHABILITY is 2 bit if at most one edge is disappearing.*

We look for a prover-verifier-pair that takes the assumption of a disappearing edge into consideration. Considering *s-t*-reachability given initially, the property only changes if the disappearing edge  $d$  is on the path from  $s$  to  $t$  and there exists no other path from  $s$  to  $t$  that does not include  $d$ . Therefore, a prover considering one disappearing edge should mark two distinct paths wherever possible to still prove the property even if one of these paths breaks due to the disappearing edge. With *distinct paths* we thereby refer to paths that do not share any vertices except for the first and last.

More precisely, any *s-t*-path  $P$  can be segmented in *critical path segments* and *uncritical path segments*, where *critical path segments* are path segments that are also part of all other paths from  $s$  to  $t$  and *uncritical path segments* are the remaining segments of  $P$ . A prover can now differentiate between *critical-path-vertices*, *uncritical-path-vertices*

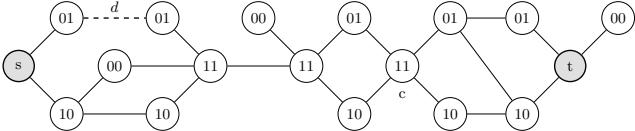


Figure 1: Example graph showing a possible 2 bit proof labeling to verify  $s$ - $t$ -reachability considering that at most one edge might disappear, e.g.,  $d$ . Here, non-path-vertices are labeled 00, critical-path-vertices are labeled 11 and uncritical-path-vertices 10 or 01, respectively, marking the two possible uncritical path segments chosen by the prover. Note that a critical path segment can consist of a single vertex.

and *non-path-vertices*. Thereby, *critical-path-vertices* are all vertices that are on every path from  $s$  to  $t$ , i.e., all vertices on critical path segments including the endpoints of the critical path segments. *Uncritical-path-vertices* are vertices chosen by the prover to mark two distinct paths for every uncritical path segment, whereas *non-path-vertices* are the remaining vertices in the graph. A simple coding of these vertex classes generates the labeling the prover can put on the graph  $G_i \in Y$ .

A verifier can then read neighboring labels together with the first inconsistency reporting bit on every vertex  $v \in V$  of the altered graph  $G_a$  and verify

- if the proof-labeling is a correct proof-labeling indicating  $s$ - $t$ -reachability before the graph was altered by checking if the amount of each label class in  $M(v)$  aligns with a possible proof labeling of a YES-instance
- if the disappearing edge (if there is any) disappeared on a critical path segment, on an uncritical path segment or not on any path specified by the prover by checking the vertex label  $l(v)$  and the first inconsistency reporting bit on the vertex
- if  $s$ - $t$ -reachability is given in the altered graph by taking the first two points into consideration

Consider Figure 1 as a possible example how such a proof labeling might look like.

To prove Theorem 1 we first consider the following lemma:

**LEMMA 2.** *There exists a correct PVP with a proof size of 2 bit that is correct for  $s$ - $t$ -REACHABILITY if at most one edge  $d$  disappears.*

**PROOF.** A prover-verifier-pair  $(\mathcal{P}, \mathcal{V})$  as required is described here. Given  $G_i = (V, E_i) \in s$ - $t$ -REACHABILITY let  $C \subset V$  denote the set of all critical path vertices. Note that  $s, t \in C$ .

The prover  $\mathcal{P}$  first labels all vertices  $v_c \in C \setminus \{s, t\}$  with the label  $l(v_c) = 11$ . For a given path from  $s$  to  $t$ , each uncritical path segment  $U_i$  has a starting vertex  $u_s(U_i) \in C$  and an termination vertex  $u_t(U_i) \in C$ . For each uncritical path segment  $U_i$  the prover  $\mathcal{P}$  chooses two distinct paths  $P_1(U_i), P_2(U_i)$  from  $u_s(U_i)$  to  $u_t(U_i)$  and labels all not yet labeled vertices on  $P_1(U_i)$  with 01 and all not yet labeled vertices on  $P_2(U_i)$  with 10. Such two distinct paths for uncritical path segments exist as a consequence of the definition of critical-path-vertices.  $\mathcal{P}$  then labels all remaining vertices with the label 00. See Figure 1 as an example of this proof labeling.

For the verifier  $\mathcal{V}$  consider the pseudo code in Algorithm 1.

Given  $G_a \in s$ - $t$ -REACHABILITY we show that  $\mathcal{V}$  outputs YES on all vertices if  $l$  was obtained from  $\mathcal{P}$ :

- All vertices labeled 00 output YES by default
- All vertices labeled 01 or 10 are on an uncritical path segment. Therefore they have exactly two neighbors indicating a possible  $s$ - $t$ -path or one such neighbor and an adjacent edge that disappeared. Note that for the latter  $s$ - $t$ -reachability is still given since the one and only disappearing edge disappeared on an uncritical path segment. In both cases the verifier  $\mathcal{V}$  will output YES.

- All vertices labeled 11 are critical-path-vertices that have either

1. two critical-path-vertices as neighbors
2. one critical-path-vertex as neighbor and they are an end point of an uncritical path segment or
3. no critical-path-vertex as neighbor, meaning that they are in between two uncritical path segments (see vertex  $c$  in Figure 1 as an example)

If no adjacent edge disappeared this can be summarized to  $y = 2$ , where  $y$  is the number of critical-path-vertices in  $M(v)$  plus the number of uncritical-path-vertices pairs. This holds since for every adjacent uncritical path segment there must be one neighbor labeled 01 and either one neighbor labeled 10, forming a pair, or directly another critical vertex. Furthermore the path should continue in both directions (as critical or uncritical path segment) which leads to  $y = 2$ .

If however an edge disappeared on an adjacent uncritical path, point 2 and 3 of the enumeration above have to be accounted for and are summarized to  $\{y = 1 \wedge o(01) + o(10) = 1 \wedge r_1 \text{ is set}\}$  and  $\{y = 1 \wedge o(01) + o(10) = 3 \wedge r_1 \text{ is set}\}$  respectively.

In all of these cases the verifier  $\mathcal{V}$  will output YES.

- Finally, a vertex carrying the label  $s$  or  $t$  marks the start or end of the path in question, therefore it neighbors either a critical-path-vertex or an uncritical path segment. With the same argumentation as before this can be summarized to  $y = 1$  if no adjacent edge disappears or  $y = 0 \wedge o(01) + o(10) = 1 \wedge r_1 \text{ is set}$  if an adjacent uncritical path edge disappears. In both cases the verifier  $\mathcal{V}$  will output YES.

Therefore  $\mathcal{V}$  will generate the output YES on all vertices of  $G_a \in s$ - $t$ -REACHABILITY if  $l$  was obtained from  $\mathcal{P}$ .

What is left to show is that given  $G_a \notin s$ - $t$ -REACHABILITY,  $\mathcal{V}$  will compute NO as output on at least one vertex.

Consider for the sake of contradiction that there exists a graph  $G_a \notin s$ - $t$ -REACHABILITY that triggers  $\mathcal{V}$  to compute YES on every vertex.

Such a graph must have a vertex  $s$  due to the problem setting. Since  $o(l) \geq 0$  for all labelings  $l$ , Algorithm 1 requires  $s$  to have either

- Case 1: a vertex labeled as critical-path-vertex as neighbor. In this case a vertex labeled 11 since the only other possibility ( $t$ ) would lead already to a contradiction.
- Case 2: a vertex labeled 01 and a vertex labeled 10 as neighbor. Or
- Case 3: a vertex labeled 01 or 10 as neighbor and an adjacent disappearing edge ( $r_1$  set)

We consider these cases separately:

---

**Algorithm 1:** Pseudo code of verifier  $\mathcal{V}$  checking  $s$ - $t$ -reachability if at most one edge is disappearing.  $G_a$  denotes the altered graph,  $l(v)$  the label on vertex  $v$ ,  $M(v)$  the set of messages received by  $v$ ,  $r_1(v)$  the first inconsistency reporting bit (reporting an adjacent disappearing edge) and  $o(\cdot)$  the number of occurrences a given label has in  $M(v)$ .

```

foreach  $v \in V(G_a)$  do
  Input:  $l(v), M(v), r_1(v) \in R(v)$ 
   $o(\cdot) \equiv o_{M(v)}(\cdot)$ 
  if  $l(v) = 00$  then
    | Output: YES
  if  $l(v) = 01$  OR  $l(v) = 10$  then
    | // Define  $x$  as the number of path vertices in
      | the neighborhood
    |  $x = o(11) + o(s) + o(t) + o(l(v))$ 
    | if  $x = 2$  OR  $x = 1 \wedge r_1$  is set then
    |   | Output: YES
    | else
    |   | Output: No
  // Set  $y$  as the number of critical-path-vertices in
  // the neighborhood
   $y = o(11) + o(s) + o(t)$ 
  // A pair{01, 10} is a continuation of the path
  foreach pair {01, 10}  $\in M(v)$  do
    |  $y = y + 1$ 
    if  $l(v) = 11$  then
      | if  $y = 2$  OR
        |  $y = 1 \wedge o(01) + o(10) = 1 \wedge r_1$  is set OR
        |  $y = 1 \wedge o(01) + o(10) = 3 \wedge r_1$  is set then
        |   | Output: YES
      | else
      |   | Output: No
    if  $l(v) \in \{s, t\}$  then
      | if  $y = 1$  OR
        |  $y = 0 \wedge o(01) + o(10) = 1 \wedge r_1$  is set then
        |   | Output: YES
      | else
      |   | Output: No
    if  $l(v) \notin \{s, t, 00, 01, 10, 11\}$  then
      | Output: No
  
```

---

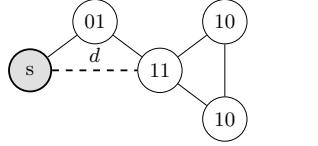
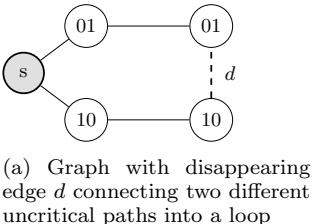


Figure 2: Example graphs to show that one side might end in a loop. Here  $d$  denotes a disappearing edge. In these graphs, the verifier  $\mathcal{V}$  would compute YES for every vertex. However these graphs do not contain  $t$  which contradicts the model assumption. Since there is at most one edge disappearing, any subgraph containing  $t$  must be connected to  $s$  or grow to infinity if it had to output YES on all vertices as well.

Case 1: For the vertex  $v$  labeled 11,  $y \geq 1$  since a critical vertex ( $s$ ) is its neighbor. Therefore Algorithm 1 requires either

- $y = 2$  which requires a next vertex to have a label 11 having  $v$  as neighbor (which is equivalent to case 1) or a pair{01, 10} as next neighbors(which is equivalent to case 2). Or
- $y = 1$  and a vertex labeled 01 or 10 as additional neighbor and an adjacent disappearing edge. This is equivalent to case 3.

Case 2: A vertex  $u$  carrying a label  $l \in \{01, 10\}$  outputs YES if

- two of its neighbors report to be either a critical-path-vertex or to have the same label as  $u$  has. This ensures a continuation of the uncritical path up to a next critical-path-vertex, considering that  $G_a$  is finite. This leads to case 4. Or
- only one neighbor reports to be critical-path-vertex or to have the same label as  $u$  and  $u$  has an adjacent disappearing edge. This can happen to one of the two paths initiated by the first two vertices labeled 01 and 10 or the disappearing edge disappeared between a vertex labeled 01 and a vertex labeled 10. If only one path gets interrupted by a disappearing edge, the other path is guaranteed to lead to a next critical-path-vertex (case 4). If the disappearing edge connects both path ends (see Figure 2a for illustration), we have a loop requiring a disappearing edge (case 5).

Case 3: Similar to case 2 here the disappearing edge is already at the start of one of the two uncritical paths. Since there is at most one disappearing edge, Algorithm 1 ensures a continuation of the other path to a next critical-path-vertex (case 4) or that the path ends up on the other side of the edge that disappeared, forming a loop (case 5).

Case 4: In this case we arrive at a critical-path-vertex  $v_c$  that must carry the label 11 (since  $t$  is not an option) and has at least one neighbor carrying a label  $l \in \{01, 10\}$ . Algorithm 1 requires here for a YES either

- $y = 1$  leading to a next critical-path-vertex (case 1),
- $y = 2$  from two pairs{01, 10} leading to two next uncritical path vertices with distinct labels (case 2) or
- $o(01) + o(10) = 3 \wedge r_1$  is set. If the two next path neighbors have distinct labels, we are again at case 2. However if they have the same label, this is a case where the path might end in a loop (see Figure 2b for illustration). Note that this case requires an adjacent disappearing edge. Therefore we are again at case 5.

Case 5: In this case we end up in a loop that somewhere required a disappearing edge (see Figure 2 for two examples). Since there is at most one disappearing edge and the whole path argumentation can also be started from  $t$  instead of  $s$ , only one side can end in this case.

Therefore, since every case leads to another case, at least from one side the path continues to infinity which contradicts the assumption of a finite graph  $G_a$ .

This concludes the proof of Lemma 2.  $\square$

**REMARK 3.** Note that this PVP does not deliver the correct output for each individual connected component separately, since on one side the path might end up in a loop

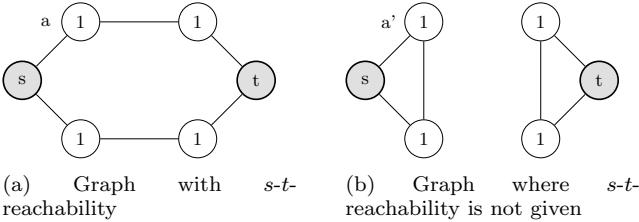


Figure 3: Example graphs showing that extending the idea of marking two paths from  $s$  to  $t$  with the same label each will not work: The vertices  $a$  and  $a'$  get exactly the same information, therefore it is indistinguishable for a verifier whether the vertices are on a loop or an  $s$ - $t$ -path.

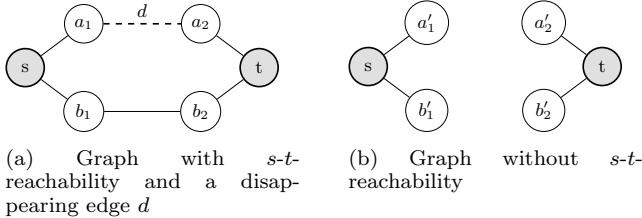


Figure 4: Example graphs to show why the ability to sense an adjacent disappearing edge is necessary. If a vertex cannot sense an adjacent disappearing edge, vertices  $a'_1$ ,  $a'_2$ ,  $b'_1$  and  $b'_2$  would all get the same information as  $a_1$  and  $a_2$  and therefore would be indistinguishable for a verifier.

as shown in Figure 2. Interestingly, we can show that no such PVP exists that can consider each individual connected component separately. The proof details are deferred to the full version of this article.

LEMMA 4. *The proof size for  $s$ - $t$ -REACHABILITY considering that one edge  $d$  might disappear is at least 2 bits.*

Combining Lemma 2 and Lemma 4 yields Theorem 1. Again, due to space restrictions, the proof details of Lemma 4 are deferred to the full version of this article. We refer to Figure 3 for an example why extending the standard checkability idea without failing edges does not suffice.

### 3.2 One Appearing Edge

In this subsection we look at what happens if at most one edge appears, i.e.,  $E(G_a) = E(G_i) \vee E(G_j) \cup \{a\}$  where  $a$  can appear between any two vertices  $u, v \in V$ . We focus on the non-trivial case of  $s$ - $t$ -reachability not given initially, first explaining the idea and then proving the following theorem:

THEOREM 5. *There exists a PVP with a proof size in  $\mathcal{O}(\log n)$  bit that is correct for  $s$ - $t$ -REACHABILITY if at most one edge  $a$  appears.*

If  $s$ - $t$ -reachability is not given initially,  $s$  and  $t$  must be located in two different connected components of the graph. An appearing edge  $a$  can make the altered graph  $G_a \in s$ - $t$ -REACHABILITY if and only if it connects these two connected components. Since  $a$  can appear between any two vertices, every vertex must be able to verify locally

1. if it is connected to  $s$ ,  $t$  or neither  $s$  nor  $t$

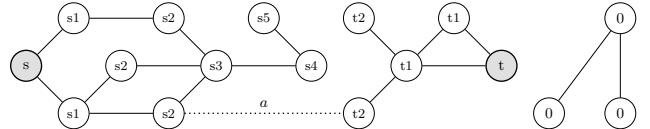


Figure 5: Example graph showing a possible  $\mathcal{O}(\log n)$  bit proof labeling to verify  $s$ - $t$ -reachability considering that at most one edge might appear, e.g.,  $a$ . Here, proof labels consist of  $s$  or  $t$  and the distance to  $s$  or  $t$ . For vertices not connected to either  $s$  nor  $t$  the label 0 is used.

2. if it got an adjacent appearing edge connecting it to a different connected component, possibly resulting in  $s$ - $t$ -reachability
3. if an edge appeared somewhere else in the graph, possibly resulting in  $s$ - $t$ -reachability

The second and third point can be checked by the second and third inconsistency reporting bits.

The first point can be assured by introducing a labeling composed of a *prefix* and a *distance*. Thereby the *prefix* of each label  $l(v)$  indicates whether  $v$  is connected to  $s$  or to  $t$  and the *distance* indicates the minimal distance to  $s$  or  $t$ , respectively. Since we focus on the non-trivial case, a vertex cannot be connected to both  $s$  and  $t$  at the time of labeling. If a vertex is connected to neither  $s$  nor  $t$ , a simple label 0 can be used.

To decode the prefix of such a labeling we introduce a *prefix function*  $p : \{0, 1\}^* \rightarrow \{s, t, 0\}$  that takes a label  $l_v$  and computes whether the label contains a prefix referring to  $s$  ( $p(l_v) = s$ ), a prefix referring to  $t$  ( $p(l_v) = t$ ) or no prefix ( $p(l_v) = 0$ ). Note that  $p(s) = s$  and  $p(t) = t$ .

To decode the distance of such a labeling we introduce a *distance function*  $d : \{0, 1\}^* \rightarrow \mathbb{N}_0 \cup \{\infty\}$  that takes a label  $l_v$  and returns

- 0, if  $l_v \in \{s, t\}$
- the distance, if  $l_v$  contains a distance  $> 0$  or
- $\infty$  if  $l_v$  does not contain a distance or an invalid (e.g.,  $\leq 0$ ) distance and  $l_v$  is neither  $s$  nor  $t$ .

PROOF. For  $G_i = (V, E_i) \notin s$ - $t$ -REACHABILITY we present a PVP  $(\mathcal{P}, \mathcal{V})$  as required. Let  $S \subset V$  denote the set of all vertices  $v_s$  that  $s$  can reach, i.e., for which there exists a path from  $s$  to  $v_s$ . Similarly let  $T \subset V$  denote the set of all vertices that can be reached by  $t$ .

The prover  $\mathcal{P}$  labels all vertices  $v_s \in S$  with the label  $s$  concatenated with the shortest distance to  $s$ . Similarly  $\mathcal{P}$  labels all  $v_t \in T$  with  $t$  and the shortest distance to  $t$ . All remaining vertices  $v \in V \setminus (S \cup T)$  are labeled with the label 0. Since any distance is strictly smaller than the number of vertices  $n$  and the labels  $s$  and  $t$  are of constant size, this proof size is in  $\mathcal{O}(\log n)$ . An illustration of such a labeling can be found in Figure 5.

For the verifier  $\mathcal{V}$  consider the pseudo code in Algorithm 2.

Given  $G_a \in s$ - $t$ -REACHABILITY we show that  $\mathcal{V}$  outputs YES on all vertices if  $l$  was obtained from  $\mathcal{P}$ . Note that in this case an edge  $a$  must have appeared connecting the two connected components containing  $s$  and  $t$ , respectively. Therefore the third inconsistency reporting bit  $r_3(v) \in R(v)$  is set for all vertices  $v \in V$ . Looking at Algorithm 2 and given that the labeling  $l$  was obtained from  $\mathcal{P}$  we note that

- each vertex  $v_0$  labeled  $l(v_0) = 0$  outputs YES since

**Algorithm 2:** Pseudo code of verifier  $\mathcal{V}$  checking  $s$ - $t$ -reachability if at most one edge is appearing.  $G_a$  denotes the altered graph,  $l(v)$  the label on vertex  $v$ ,  $M(v)$  the set of messages received by  $v$ ,  $r_2(v)$  the second inconsistency reporting bit (reporting an adjacent appearing edge) and  $r_3(v)$  the third inconsistency reporting bit (reporting an appearing edge in the graph).  $o_{prefix}(k)$  shall denote the number of labels in  $M(v)$  carrying the prefix  $k$ , where  $k \in \{s, t\}$ .  $p(\cdot)$  denotes the prefix decoding function and  $d(\cdot)$  denotes the distance decoding function.

```

foreach  $v \in V(G_a)$  do
    Input:  $l(v), M(v), r_2(v) \in R(v), r_3(v) \in R(v)$ 
    if  $r_3(v)$  is not set then
        Output: No
        if  $l(v) = 0$  then
            if  $r_2(v)$  is set then
                Output: No
            else
                Output: Yes
        // Decode  $l(v)$  into prefix  $p_v \in \{s, t\}$  and distance  $d_v$ .
         $p_v = p(l(v))$ 
         $d_v = d(l(v))$ 
        if  $p_v = 0$  OR  $d_v = \infty$  then
            Output: No
        if  $l(v) \in \{s, t\} \wedge (s \in M(v) \text{ OR } t \in M(v))$  then
            Output: Yes
        if  $d_v = 0 \wedge \nexists l_0 \in M(v) : d(l_0) = 0$  OR
             $\exists l_1 \in M(v) : d(l_1) = d_v - 1$  then
            if  $r_2(v)$  is set then
                if  $o_{prefix}(p_v) = |M(v)| - 1 \wedge$ 
                     $\exists l \in M(v) : p(l) \in \{s, t\} \wedge p(l) \neq p_v$  then
                        Output: Yes
                else
                    if  $o_{prefix}(p_v) = |M(v)|$  then
                        Output: Yes
            if non of the cases above then
                Output: No

```

the the appearing edge  $a$  connects the connected component containing  $s$  to the connected component containing  $t$  and therefore no vertex labeled 0 can have an adjacent appearing edge

- vertices  $s$  and  $t$  have a distance  $d_v = 0$  and no neighbor with distance 0 unless the edge appeared between  $s$  and  $t$  (which is separately handled in the algorithm).
- each vertex  $v \in \{S \cup T\} \setminus \{s, t\}$  has at least one neighbor with distance  $d_n = d_v - 1$  where  $d_v$  denotes the distance of  $v$
- there exists a vertex  $v_{a,s} \in S$  and a vertex  $v_{a,t} \in T$  that have an adjacent appearing edge (second inconsistency reporting bit  $r_2(v) \in R(v)$  set).  $v_{a,s}$  and  $v_{a,t}$  receive exactly one prefix  $p(l) \in \{s, t\} \neq p_v$  where  $p_v$  denotes their own prefix. This prefix  $p(l)$  is the one received through the edge  $a$  that appeared. All other prefixes received from neighbors are equal to  $p_v$  since they were all in the same connected component of the

initial graph  $G_i$ . Given all this,  $\mathcal{V}$  computes YES for  $v_{a,s}$  and  $v_{a,t}$

- for each vertex  $v \in \{S \cup T\} \setminus \{v_{a,s}, v_{a,t}\}$   $r_2(v)$  is not set, since there is at most one appearing edge. All prefixes received by  $v$  are equal to its own prefix  $p_v$  since they were all in the same connected component of the initial graph  $G_i$ . Therefore  $\mathcal{V}$  will compute YES for all those vertices as well

This concludes that  $\mathcal{V}$  will compute YES for every vertex of an altered graph  $G_a \in s$ - $t$ -REACHABILITY if the labeling  $l$  was obtained from  $\mathcal{P}$ .

What is left to show is that given  $G_a \notin s$ - $t$ -REACHABILITY,  $\mathcal{V}$  will compute NO as output on at least one vertex.

Assume for the sake of contradiction that there exists a graph  $G_a \notin s$ - $t$ -REACHABILITY for which  $\mathcal{V}$  computes YES on all vertices.

Considering Algorithm 2 and the case of no edge appearing ( $r_3(v)$  is not set for any  $v \in V$ ),  $\mathcal{V}$  would compute NO for all vertices. Therefore  $G_a$  must have an edge  $a$  that appeared.

We further note that for any vertex  $v$  carrying a prefix  $p_v$  the following must hold:

1. if  $r_2(v)$  is not set, all neighbors of  $v$  must carry the same prefix  $p_v$  since  $\mathcal{V}$  requires  $o_{prefix}(p_v) = |M(v)|$  to output YES
2. if  $r_2(v)$  is set, all neighbors but one must carry the same prefix  $p_v$  and the remaining neighbor must have a label  $l$  with prefix  $p(l) \in \{s, t\}$  not equal to  $p_v$ . Since there are only the prefixes  $s$  and  $t$ ,  $p(l) = t$  if  $p_v = s$  and vice versa.

From Point 1 it follows that if any vertex within a connected component carries a prefix  $p_v$  and no vertex within that connected component has an adjacent appearing edge, all vertices of the connected component must carry the prefix  $p_v$ . From Point 2 it follows that if a vertex  $v$  has an adjacent appearing edge,  $v$  must be connected to a vertex carrying a different prefix  $p(l) \in \{s, t\}$ . Thus, if the appearing edge  $a$  is adjacent to any vertex carrying a prefix, there must be a connection between a connected component carrying only prefixes  $s$  and a connected component with only prefixes  $t$ .

Next, consider that Algorithm 2 requires every vertex  $v$  with  $l(v) \neq 0$  to have a prefix  $p_v \in \{s, t\}$  and a distance  $d_v \neq \infty$ . We note that distances assigned by the labeling  $l(\cdot)$  must be strictly  $> 0$  for vertices  $v \notin \{s, t\}$  since otherwise  $d(\cdot)$  would decode them as  $\infty$  which would lead  $\mathcal{V}$  to output NO. Furthermore, each vertex  $v$  with  $d_v > 0$  requires a neighbor  $u$  with distance  $d_u = d_v - 1$ , especially, vertices having a distance 1 require either  $s$  or  $t$  in their neighborhood.

Therefore, since  $s$  and  $t$  are unique, there can only be one connected component carrying the prefixes  $s$  containing  $s$  itself and one connected component carrying the prefixes  $t$  containing  $t$  itself. None of those can have the appearing edge  $a$  adjacent to any of their vertices since this would, as shown earlier, request a connection between the two components, thereby connecting  $s$  and  $t$ .

The only accepted label not carrying a prefix is 0. However, if an appearing edge  $a$  were adjacent to a vertex  $v$  carrying the label  $l(v) = 0$ ,  $\mathcal{V}$  would compute NO as output for  $v$ . Therefore, since there has to be an appearing edge  $a$  but no vertex in  $G_a$  can have an adjacent appearing edge, we have a contradiction proving our initial claim.  $\square$

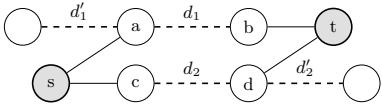


Figure 6: Graph showing that  $s$ - $t$ -reachability cannot be verified in a local manner if there might be two edges disappearing. Here, vertex  $a$  can sense an adjacent disappearing edge  $d_1$ , but cannot receive enough information within one round of communication to decide if a second disappearing edge disappeared at a location  $d_2$  or  $d'_2$ . Since it might be  $d'_2$ , a verifier on  $a$  has to output YES. The same applies however to  $b$  as well as to  $c$  and  $d$ , where  $c$  and  $d$  cannot know whether  $d_1$  or  $d'_1$  disappeared. This leads all vertices to output YES even though  $s$ - $t$ -reachability might not be given.

### 3.3 Multiple Inconsistencies

In this subsection we present the challenges in checking  $s$ - $t$ -reachability when considering multiple inconsistencies. We will first look at multiple disappearing edges, then look at multiple appearing edges, and end with combinations of disappearing and appearing edges. Due to space constraints, the technical proofs of this subsection are deferred to the full version of this article.

#### 3.3.1 Multiple Disappearing Edges

We consider the initial graph  $G_i \in s$ - $t$ -REACHABILITY and ask whether there is a correct PVP to verify if  $G_a \in s$ - $t$ -REACHABILITY with  $E(G_a) = E(G_i) \setminus D$  where  $D$  denotes the set of disappearing edges. Sadly, the answer is no, since edges might disappear anywhere in the graph and the verifier on a vertex  $v$  can only check the local neighborhood of  $v$ . We refer to Figure 6 for an illustration of the deferred proof.

**THEOREM 6.** *There exists no correct PVP for  $s$ - $t$ -REACHABILITY as specified in our model that can consider more than one disappearing edge.*

#### 3.3.2 Multiple Appearing Edges

We consider the initial graph  $G_i \notin s$ - $t$ -REACHABILITY and ask whether there is a correct PVP to verify if  $G_a \in s$ - $t$ -REACHABILITY with  $E(G_a) = E(G_i) \cup A$  where  $A$  denotes the set of appearing edges.

As with disappearing edges in Section 3.3.1 appearing edges can appear anywhere in the graph and therefore cannot be located within the graph by a non-adjacent vertex. We refer to Figure 7 for an illustration of the deferred proof.

**THEOREM 7.** *There exists no correct PVP for  $s$ - $t$ -REACHABILITY as specified in our model that can consider more than one appearing edge.*

#### 3.3.3 Combination of Dis- and Appearing Edges

We consider now any initial graph  $G_i$  and ask whether there is a correct PVP to verify if  $G_a \in s$ - $t$ -REACHABILITY with  $E(G_a) = \{E(G_i) \cup A\} \setminus D$  where  $A$  denotes the set of appearing edges and  $D$  denotes the set of disappearing edges.

Assuming that there is at most either one edge disappearing or at most one edge appearing, a combination of the algorithms presented in Section 3.1 and Section 3.2 could

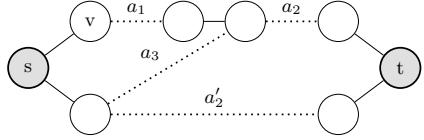


Figure 7: Graph showing that  $s$ - $t$ -reachability cannot be verified in a local manner if there might be two edges appearing. Here, vertex  $v$  can sense an adjacent appearing edge  $a_1$ , but cannot receive enough information within one round of communication to decide if a second appearing edge  $a_2$  or  $a'_2$  exists that would lead to  $s$ - $t$ -reachability. Note that even if  $v$  would get the information of another appearing edge, it could also be that the second edge that appeared is  $a_3$  (and not  $a_2$  or  $a'_2$ ), leaving  $s$  and  $t$  unconnected.

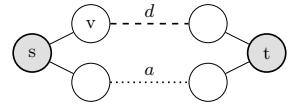


Figure 8: Graph showing that  $s$ - $t$ -reachability cannot be verified in a local manner if there might be an edge disappearing and an edge appearing. Here, vertex  $v$  can sense an adjacent disappearing edge  $d$ , but cannot receive enough information within one round of communication to decide if an appearing edge  $a$  on the other side of the graph exists that would lead to  $s$ - $t$ -reachability.

make up a PVP for  $s$ - $t$ -REACHABILITY with proof size in  $\mathcal{O}(\log n)$ . This by choosing the proof labeling according to whether the initial graph  $G_i$  is in  $s$ - $t$ -REACHABILITY or not (if it is, choose the labeling from Section 3.1, if not choose the labeling from Section 3.2) and adapt the verifier to switch to Algorithm 1 or Algorithm 2 depending on the proof labeling it finds.

However, assuming that there is at most one edge disappearing AND at most one edge appearing, a correct PVP for  $s$ - $t$ -REACHABILITY cannot be found. This due to the fact that if there is a disappearing edge  $d$  breaking the graph into two connected components, vertices adjacent to  $d$  cannot know if there is an appearing edge on the other end of the graph connecting the two components again. We refer to Figure 8 for an illustration of the deferred proof.

**THEOREM 8.** *There exists no correct PVP for  $s$ - $t$ -REACHABILITY as specified in our model that can consider a combination of at most one disappearing edge and of at most one appearing edge.*

## 4. CYCLICITY

Cyclicity, i.e., if a given graph contains edges that form a cycle, is one of the key attributes when analyzing networks. Especially for routing protocols it is important to know if the routed packet gets forwarded indefinitely in a cycle or is guaranteed to advance to a new vertex with every new step.

In the following subsections we discuss if a prover verifier pair can be found to check if a graph contains a cycle or not under the assumption of inconsistency. We will look at individual connected components of graphs. Therefore we introduce  $C(G) = (V_C, E_C)$  to be a *connected component* of a graph  $G$ , with  $V_C \subset V(G)$  and  $E_C \subset E(G)$  such that

any two vertices  $u, v \in V_C$  can reach each other. We further introduce two small changes to our model:

1. Let  $Y$  be a network property. A prover verifier pair  $(\mathcal{P}, \mathcal{V})$  is said to be *correct* for  $Y$  if the following two points hold for any connected component  $C(G_a) = (V_{C,a}, E_{C,a})$  of the altered graph  $G_a$ :
  - (a) if  $C(G_a) \in Y$  and  $l$  was obtained from  $\mathcal{P}$ , then  $\mathcal{V}$  computes YES as output for all vertices  $v \in V_{C,a}$
  - (b) if  $C(G_a) \notin Y$ , then  $\mathcal{V}$  computes NO as output for at least one vertex  $v \in V_{C,a}$ , regardless of the labeling
2. for a vertex  $v \in V(G_a)$ , the third inconsistency reporting bit  $r_3(v)$  is set if and only if there has been an appearing edge adjacent to  $v$  or any vertex  $u$  connected to  $v$ , i.e., if there has been an appearing edge adjacent to any vertex of the connected component  $C(G_a)$  to which  $v \in V_{C,a}$  belongs

1-CYCLIC we shall denote the set of all (sub-)graphs containing exactly one cycle, and CYCLIC shall denote the set of all (sub-)graphs containing at least one cycle.

## 4.1 Disappearing Edges

We first look at the case that edges might disappear. As it was with  $s-t$ -reachability in Section 3.1 we focus here on the case that the initial graph  $G_i \in \text{CYCLIC}$ .

We divide this problem further into two cases:

1. The initial graph  $G_i$  contains exactly one cycle, i.e.,  $G_i \in \text{1-CYCLIC}$
2. The initial graph  $G_i$  contains at least one cycle, i.e.,  $G_i \in \text{CYCLIC}$

### 4.1.1 Single Cycle

In [8] a proof size of 2 bits was shown for cyclicity checking in unaltered undirected graphs. For the case that  $G_i \in \text{1-CYCLIC}$  has at most one cycle, the PVP [8] introduces works also under the consideration of disappearing edges and the model adaptions we introduced.

This is due to the fact that if there is at most one cycle, a disappearing edge can only break the cycle or split a connected component in two connected components. Therefore each connected component  $C(G_a)$  of the altered graph  $G_a$  can be seen as an unaltered graph and verified individually, if it contains a cycle or not.

**THEOREM 9.** *The proof size for 1-CYCLIC is 2 bit for any number  $k \leq |E_i|$  of edges disappearing.  $|E_i|$  denotes the number of edges in the initial graph.*

[8] proves that the proof size for CYCLIC in undirected graphs is at least 2 bits. Since their model does not consider any inconsistencies and is therefore stronger, this lower bound holds also in our model. To prove Theorem 9 we are left to prove the following lemma:

**LEMMA 10.** *There exists a correct PVP with a proof size of 2 bit that is correct for 1-CYCLIC if any number  $k \leq |E_i|$  of edges might disappear.  $|E_i|$  denotes the number of edges in the initial graph.*

Due to space constraints, we defer the proof to the full version of this article. See Figure 9 for an example of the proof labeling used.

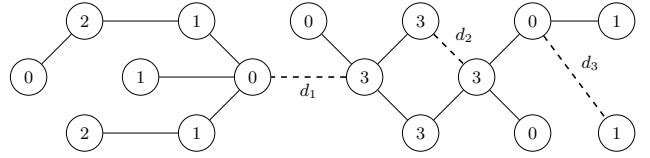


Figure 9: Example graph showing the 2 bit proof labeling introduced by [8] to verify cyclicity. We take an analogous PVP showing that it is correct for 1-CYCLIC considering that edges might disappear, e.g.,  $d_1$ ,  $d_2$  and/or  $d_3$ . Here, vertices on the cycle are labeled 3 and vertices not on the cycle are labeled with their distance modulo 3 to the root of their subtree, where the root is a vertex next to the cycle.

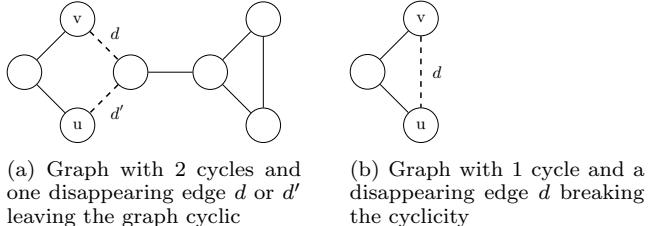


Figure 10: Example graphs to show why no PVP can be correct for CYCLIC if one edge might disappear. A prover would have to give  $v$  in Figure 10a a label for which the verifier outputs YES with adjacent disappearing edge  $d$  since there is a connection through  $u$  to the other cycle. The same holds for the label  $u$  gets since if  $d'$  disappears there is a connection through  $v$  to the other cycle. Such a labeling would however output YES on all vertices in Figure 10b.

### 4.1.2 Multiple Cycles

The problem gets much more complicated when more than one cycle can exist in the graph. For this we consider now the case where the initial graph  $G_i \in \text{CYCLIC}$  can have any amount of cycles and at most one edge  $d$  might disappear.

We take as example a connected initial graph  $G_i$  that has two cycles and note that the graph is still cyclic even if one of the cycles breaks due to the disappearing edge. Therefore, vertices on each cycle must get some information from the labeling that there exists another cycle in the graph. This however leads to an insolvable problem in our model as we will show. More precisely:

**THEOREM 11.** *There exists no correct PVP for CYCLIC as specified in our model that can consider one disappearing edge.*

**PROOF.** Assume for the sake of contradiction that there exists a correct PVP  $(\mathcal{P}, \mathcal{V})$  as required. Consider the example graphs shown in Figure 10.

Given the initial graph  $G_{i,(a)}$  in Figure 10a the prover  $\mathcal{P}$  would have to give the vertex  $v$  a label  $l(v)$ , for which the verifier  $\mathcal{V}$  computes YES if it senses an adjacent disappearing edge  $d$ , i.e., if the first inconsistency reporting bit  $r_1(v)$  is set. Such a label  $l(v)$  is needed since if  $d$  disappears in Figure 10a there is still a path from  $v$  through  $u$  to the other cycle in the graph. (Note that at most one edge disappears, so either  $d$  or  $d'$ ). However, a similar label  $l(u)$  is needed for the vertex  $u$  since  $d'$  might disappear instead of  $d$  and  $u$  would then still have a path through  $v$  to the other cycle. Therefore  $v$  and  $u$  must have labels for which  $\mathcal{V}$  computes YES if there

is an adjacent disappearing edge. This however would be a labeling for which  $\mathcal{V}$  computes YES on every vertex in Figure 10b. This contradicts that  $\mathcal{V}$  should compute NO for at least one vertex for any connected component  $C(G_a) \notin \text{CYCLIC}$ , regardless of the labeling.  $\square$

Given that there is no correct PVP for CYCLIC that can consider one disappearing edge, the question about a correct PVP for CYCLIC that can consider multiple disappearing edges becomes redundant.

## 4.2 One Appearing Edge

In this subsection we take a look at how cyclicity might change if at most one edge  $a$  appears, i.e.,  $E(G_a) = E(G_i) \cup E(G_i) \cup \{a\}$  where  $a$  can appear between any two vertices  $u, v \in V$ . Before we prove or disprove a PVP, note the following facts:

- A connected component containing a cycle cannot become acyclic through an appearing edge
- A connected component  $C_1(G_i) = (V_{C1}, E_{C1})$  containing no cycle can only get cyclic if it has an appearing edge adjacent to one of its vertices and either the appearing edge connects  $C_1(G_i)$  to another connected component  $C_2(G_i)$  containing a cycle or both end vertices  $u$  and  $v$  of the appearing edge are within the connected component, i.e.  $u, v \in V_{C1}$ .
- if both end vertices  $u$  and  $v$  of the appearing edge are within the connected component,  $C_1(G_a)$  is for sure cyclic. This holds since every acyclic connected component  $C = (V, E)$  has to have exactly the minimal number of edges  $|E| = |V| - 1$  to be connected and acyclic. Any additional edge closes a cycle.

Given these facts we note that a vertex  $v$  on one side of the appearing edge must have the ability to distinguish between initially separated connected component to know, whether  $u$  on the other side of the appearing edge belongs to the same connected component or not.

Since this cannot be achieved through a potentially random labeling we allow for *IDs* in this subsection. Thereby an *ID* is a unique identifier every vertex  $v \in V(G_i)$  of the initial graph gets attached to itself, where the uniqueness is guaranteed by the model. Further, we adjust our model such that the set of received messages  $M(v)$  on vertex  $v$  not only includes labels, but also the IDs of each neighbor of  $v$ .

By adding unique identifiers as described to the model, we can now show the existence of a PVP. The technical proof is deferred to the full version of this article.

**THEOREM 12.** *There exists a PVP, using unique identifiers, with a proof size in  $\mathcal{O}(n^2 \log n)$  bit that is correct for CYCLIC if at most one edge  $a$  appears.*

## 4.3 Multiple Inconsistencies

In this subsection we focus on PVPs for cyclicity considering more than one inconsistency. We already looked at multiple disappearing edges in Section 4.1. Therefore, it is left to discuss multiple appearing edges and combinations of disappearing and appearing edges.

### 4.3.1 Multiple Appearing Edges

We consider the initial graph  $G_i$  that is a set of connected components and ask whether there is a correct PVP that

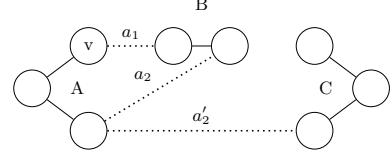


Figure 11: Graph showing that cyclicity cannot be verified in a local manner if there might be two edges appearing.  $A, B$  and  $C$  denote the three acyclic connected components of the initial graph. Here, vertex  $v$  can sense an adjacent appearing edge  $a_1$ , but cannot receive enough information within one round of communication to decide if a second appearing edge  $a_2$  exists that would lead to cyclicity. Even if  $v$  would get the information of another appearing edge, it could also be that the second edge that appeared is  $a'_2$  (and not  $a_2$ ), leaving the resulting connected component acyclic.

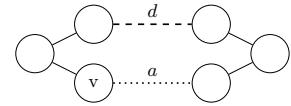


Figure 12: Graph showing that cyclicity cannot be verified in a local manner if there is an edge appearing and there might be an edge disappearing.  $v$  can sense an adjacent appearing edge  $a$ , but cannot receive the information to decide if a disappearing edge  $d$  on the other side of the graph exists that would leave the graph acyclic.

considers multiple appearing edges and verifies for each connected component  $C(G_a)$  of the altered graph  $G_a$  individually, if  $C(G_a) \in \text{CYCLIC}$ . This, having the set of edges in the altered graph  $E(G_a) = E(G_i) \cup A$  where  $A$  denotes the set of appearing edges. Remember that appearing edges can appear anywhere in the graph and therefore cannot be located within the graph by a non-adjacent vertex. We refer to Figure 11 for an illustration of the deferred proof.

**THEOREM 13.** *There exists no correct PVP for CYCLIC as specified in our model that can consider more than one appearing edge.*

### 4.3.2 Combination of Dis- and Appearing Edges

Since there is no correct PVP considering disappearing edges if the graph contains more than one cycle (see Section 4.1.2) and appearing edges tend to form cycles, the question about a correct PVP for cyclicity that considers a combination of dis- and appearing edges is rather complex.

However if we focus on at most one edge disappearing and at most one edge appearing, we can already show that no correct PVP for 1-CYCLIC can be found. This due to the fact that if there is an appearing edge  $a$  forming a cycle, vertices adjacent to  $a$  cannot know if there is a disappearing edge on the other end of the graph breaking the cycle again. We refer to Figure 12 for an illustration of the deferred proof.

**THEOREM 14.** *There exists no correct PVP for 1-CYCLIC as specified in our model that can consider more than one inconsistency of any type if at least one edge is appearing.*

## 5 RELATED WORK

The concept of local checkability has a rich history, starting with the seminal paper of Naor and Stockmeyer [18]:

They coined the term *Locally Checkable Labelings (LCL)*, checking labels locally in a constant number of rounds.

Further models for local checkability have been introduced since then, extending and building upon the ideas of [18]. We now cover the four ones closest related to our work:

Göös and Suomela [14, 15] allowed  $f(n)$  bits of additional information on the vertices introducing *Locally Checkable Proofs (LCP)* as an addition to Locally Checkable Labelings ( $f(n) = 0$  being LCL). They investigated on distributed local decision problems: which proof size  $f(n)$  is needed such that all vertices output YES on a YES-instance and at least one vertex outputs NO on any invalid proof or NO-instance.

With *Proof Labeling Schemes* Korman et al. [16] restricted the LCP model to allow for one communication round only. Thus, the verification of the proof labels are computed from each vertex' immediate neighborhood. Super-constant communication rounds have also been studied in [3].

Both the models of [14] and [16] also investigate the use of unique identifiers, however a distinction to our work is that they always assume a port numbering to be given. Further studies on more universal identifiers were performed in [12]. Cf. also [7] for a generalization, coined local hierarchy *LH*.

Fraigniaud et al. [11] took a different approach by allowing multiple communication rounds but separating the proof labels from vertex identifiers. They distributed the labels by having only the graph structure without identifiers given, but the verifier is aware of the identifiers. Unlike Göös and Suomela [14], and Korman et al. [16], Fraigniaud et al. studied the impact of randomization on local checkability as well, also regarding reduced proof sizes [2, 10, 13].

Most related to this work is the article of Foerster et al. [8, 9], which uses a combination of the above as model. By having a *Prover P* distributing proof-labels and a *Verifier V* checking the labels after one communication round, they introduce the *Prover-Verifier-Pairs (PVPs)* we adapted for this work. These have *no strings attached* meaning that they rely on neither identifiers nor port numbers of any sort to make decision. Unlike [8], our work focuses on undirected graphs, but introduces inconsistencies to the network graph.

For a recent and encompassing overview on distributed decisions beyond the works discussed above we refer to the survey of Feuilloley and Fraigniaud [6]. A further field related to local checkability is property testing, cf. [5].

We are not aware of work that connects local checkability with dynamic networks. Dynamic networks are in the focus of research for a long time [1], especially since the rise of peer-to-peer computing, but their study in the local model [4, 17] is more recent.

## 6. CONCLUDING REMARKS

We discussed the applicability of local checkability to dynamic networks, where edges might appear or disappear when verifying network properties. To the best of our knowledge, our work is the first that extends local checking to dynamic network changes. As thus, we focused on (im-)possibility results – showing that there are cases where quite simple schemes suffice, but that problems which seem equally “easy” at first glance can also allow for no local solution at all, with the occasional corrective supplied by unique identifiers beyond the possibilities of a simple prover-verifier-pair. We note that the restriction to one communication round does not fundamentally change the theoretical power of a PVP in our case: The provided counter-examples can be

extended by replacing edges with paths of nodes of length  $k$ , for any constant  $k$ , s.t. problems in NLC stay in NLC. We believe local checkability in dynamic networks to be an intriguing field of study, as it extends the fascinating theory of local verification towards the realm of topology changes.

## Acknowledgments

We would like to thank the anonymous reviewers of ICDCN 2017 for their helpful comments on our submitted manuscript.

## 7. REFERENCES

- [1] B. Awerbuch, Y. Mansour, and N. Shavit. Polynomial end-to-end communication. In *FOCS*, 1989.
- [2] M. Baruch, P. Fraigniaud, and B. Patt-Shamir. Randomized proof-labeling schemes. In *PODC*, 2015.
- [3] M. Baruch, R. Ostrovsky, and W. Rosenbaum. Brief announcement: Space-time tradeoffs for distributed verification. In *PODC*, 2016.
- [4] R. O. Bischoff and R. Wattenhofer. Information Dissemination in Highly Dynamic Graphs. In *DIALM-POMC*, 2005.
- [5] K. Censor-Hillel, E. Fischer, G. Schwartzman, and Y. Vasudev. Fast distributed algorithms for testing graph properties. *CoRR*, abs/1602.03718, 2016.
- [6] L. Feuilloley and P. Fraigniaud. Survey of distributed decision. *CoRR*, abs/1606.04434, 2016.
- [7] L. Feuilloley, P. Fraigniaud, and J. Hirvonen. A hierarchy of local decision. *ICALP*, 2016.
- [8] K.-T. Foerster, T. Luedi, J. Seidel, and R. Wattenhofer. Local Checkability, No Strings Attached. In *ICDCN*, 2016.
- [9] K.-T. Foerster, T. Luedi, J. Seidel, and R. Wattenhofer. Local Checkability, No Strings Attached: (A)cyclicity, Reachability, Loop Free Updates in SDNs. *Theor. Comput. Sci.*, To appear.
- [10] P. Fraigniaud, M. Göös, A. Korman, M. Parter, and D. Peleg. Randomized distributed decision. *Distributed Computing*, 27(6):419–434, 2014.
- [11] P. Fraigniaud, M. Göös, A. Korman, and J. Suomela. What can be decided locally without identifiers? In *PODC*, 2013.
- [12] P. Fraigniaud, J. Hirvonen, and J. Suomela. Node labels in local decision. In *SIROCCO*, 2015.
- [13] P. Fraigniaud, A. Korman, and D. Peleg. Towards a complexity theory for local distributed computing. *J. ACM*, 60c(5):35, 2013.
- [14] M. Göös and J. Suomela. Locally checkable proofs. In *PODC*, 2011.
- [15] M. Göös and J. Suomela. Locally checkable proofs. *Theory of Computing*, To appear.
- [16] A. Korman, S. Kutten, and D. Peleg. Proof labeling schemes. *Distributed Computing*, 22(4):215–233, 2010.
- [17] F. Kuhn, N. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *STOC*, 2010.
- [18] M. Naor and L. J. Stockmeyer. What can be computed locally? In *STOC*, 1993.
- [19] S. Schmid and J. Suomela. Exploiting locality in distributed SDN control. In *HotSDN*, 2013.
- [20] N. Shelly, B. Tschaen, K.-T. Foerster, M. A. Chang, T. Benson, and L. Vanbever. Destroying networks for fun (and profit). In *HotNets*, 2015.

# Distributed Computation of Mixing Time

Anisur Rahaman Molla \*  
 ACM Unit  
 Indian Statistical Institute  
 Kolkata 700108, India  
 anisurpm@gmail.com

Gopal Pandurangan †  
 Department of Computer Science  
 University of Houston  
 Houston, Texas 77204, USA  
 gopalpandurangan@gmail.com

## ABSTRACT

The mixing time of a graph is an important metric, which is not only useful in analyzing connectivity and expansion properties of the network, but also serves as a key parameter in designing efficient algorithms. We present an efficient distributed algorithm for computing the mixing time of undirected graphs. Our algorithm estimates the mixing time  $\tau_s$  (with respect to a source node  $s$ ) of any  $n$ -node undirected graph in  $O(\tau_s \log n)$  rounds. Our algorithm is based on random walks and require very little memory and use lightweight local computations, and work in the *CONGEST* model. Hence our algorithm is scalable under bandwidth constraints and can be an helpful building block in the design of topologically aware networks.

## Categories and Subject Descriptors

F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems; G.2.2 [Discrete Mathematics]: Graph Theory—*Graph algorithms, Network problems*

## Keywords

Distributed Algorithm, Random Walk, Mixing Time, Conductance, Spectral Properties

## 1. INTRODUCTION

Mixing time of a random walk in a graph is the time taken by a random walk to converge to the *stationary distribution* of the underlying graph. It is an important parameter which is closely related to various key graph properties such as graph expansion, spectral gap, conductance etc. Mixing time is related to the *conductance*  $\Phi$  and *spectral gap*

\*Work done while at Department of Computer Science, University of Freiburg, 79110 Freiburg, Germany. Research partially supported by ERC Grant No. 336495 (ACDC).

†Supported, in part, by NSF grants CCF-1527867, CCF-1540512, and IIS-1633720.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

*ICDCN '17, January 04 - 07, 2017, Hyderabad, India*

© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007784>

( $1 - \lambda_2$ ) of a  $n$ -node graph due to the known relations ([9]) that  $\frac{1}{1-\lambda_2} \leq \tau \leq \frac{\log n}{1-\lambda_2}$  and  $\Theta(1 - \lambda_2) \leq \Phi \leq \Theta(\sqrt{1 - \lambda_2})$ , where  $\lambda_2$  is the second largest eigenvalue of the adjacency matrix of the graph. Small mixing time means the graph has high expansion and spectral gap. Such a network supports fast random sampling (which has many applications [7]) and low-congestion routing [8]. Moreover, the spectral properties reveal a lot about the network structure [6]. Mixing time is also useful in designing efficient randomized algorithms in communication networks [1, 2, 5, 6, 11, 15].

In this paper, we focus on developing a simple and efficient distributed algorithm for computing mixing time in graphs. Given an undirected  $n$ -node network  $G$  and a source node  $s$ , our algorithm estimates the mixing time  $\tau_s$  for the source node in  $O(\tau_s \log n)$  rounds and achieves high accuracy of estimation. We note that this running time is non-trivial in the *CONGEST* model.<sup>1</sup> Moreover, our algorithm is simple, lightweight (low-cost computations within a node) and easy to implement.

Our approach crucially uses random walks. Random walks are very local and lightweight and require little index or state maintenance that makes it attractive especially to self-organizing and dynamic networks. Our approach, on a high-level, is based on efficiently performing many random walks from a particular node and computing the fraction of random walks that terminate over each node. We show that this fraction estimates the random walk probability distribution. Our approach achieves very high accuracy which is a requirement in some applications [4, 5, 11, 18]. Due to space constraints, we omit some proofs from the article which can be found in the full version [14].

We remark that we can compute the mixing time of a graph in  $O(m)$  time in the *CONGEST* model, where  $m$  is the number of edges in the graph. In the worst case, the mixing time  $\tau$  could be  $O(n^3)$  for some graph, e.g., the Lollipop graph. On the other hand, the complete graph topology can be collected to a single node (e.g., by electing a leader which takes  $O(D)$  rounds [12]) by flooding in  $O(m)$  rounds and then the node can compute the mixing time locally. The source node can compute the number of edges in the beginning (which can be done in  $O(D)$  time) and then run the two algorithms in parallel and stop when one of them terminates.

### 1.1 Distributed Computing Model

We model the communication network as an undirected, unweighted, connected graph  $G = (V, E)$ , where  $|V| = n$

<sup>1</sup>In the *LOCAL* model, all problems can be trivially solved in  $O(D)$  rounds.

and  $|E| = m$ . Every node has limited initial knowledge. Specifically, assume that each node is associated with a distinct identity number (e.g., its IP address). At the beginning of the computation, each node  $v$  accepts as input its own identity number and the identity numbers of its neighbors in  $G$ . We also assume that the number of nodes and edges i.e.,  $n$  and  $m$  (respectively) are given as inputs. (In any case, nodes can compute them easily through broadcast in  $O(D)$  time; this does not affect the asymptotic bounds of our algorithm.) The nodes are only allowed to communicate through the edges of the graph  $G$ . We assume that the communication occurs in synchronous *rounds*. We will use only small-sized messages. In particular, in each round, each node  $v$  is allowed to send a message of size  $O(\log n)$  bits through each edge  $e = (v, u)$  that is adjacent to  $v$ . The message will arrive to  $u$  at the end of the current round. This is a widely used standard model known as the *CONGEST model* to study distributed algorithms (e.g., see [17, 16]) and captures the bandwidth constraints inherent in real-world computer networks.

We focus on minimizing the *the running time*, i.e., the number of *rounds* of distributed communication. Note that the computation that is performed by the nodes locally is “free”, i.e., it does not affect the number of rounds; however, we will only perform polynomial cost computation locally (in particular, very simple computations) at any node.

## 1.2 Random Walk Preliminaries

We consider a *simple random walk* in an undirected graph: In each step the walk goes from the current node to a random neighbor i.e., from the current node  $v$ , the probability of moving to node  $u$  is  $\Pr(v, u) = 1/d(v)$  if  $(v, u) \in E$ , otherwise  $\Pr(v, u) = 0$ , where  $d(v)$  is the degree of  $v$ .

Suppose a random walk starts at some vertex  $v$  in a graph  $G$ . Let  $\mathbf{P}_0$  be the initial distribution with probability 1 at the node  $v$  and zero at all other nodes. Then we get a probability distribution  $\mathbf{P}_t$  at time  $t$  starting from the initial distribution  $\mathbf{P}_0$ . Note that we hide the starting node in the notation of the probability distribution  $\mathbf{P}_t$ . We hope that it is clear to the reader from the context. (Informally) we say that the distribution  $\mathbf{P}_r$  is stationary (or steady-state) for the graph  $G$  when no further changes on the distribution i.e.,  $\mathbf{P}_{r+t} = \mathbf{P}_r$  for  $t \geq 1$ . It is known that the stationary distribution of an undirected connected graph is a well defined quantity which is  $(\frac{d(v_1)}{2m}, \frac{d(v_2)}{2m}, \dots, \frac{d(v_n)}{2m})$ , where  $d(v_i)$  is the degree of the node  $v_i$ . We denote the stationary distribution vector by  $\pi$ , i.e.,  $\pi(v) = d(v)/2m$  for every node  $v$ . The stationary distribution of a graph is fixed irrespective of the starting node of a random walk, however, the time to reach to the stationary distribution could be different for the different starting nodes.

The *mixing time* of a random walk starting from the source node  $v$ , denoted by  $\tau_v$ , is the time (or number of steps) taken to reach to the stationary distribution of the graph. The mixing time, denoted by  $\tau$ , is the maximum mixing time among all (starting) nodes in the graph. The formal definitions are given below.

**DEFINITION 1.1.** ( $\tau_v(\epsilon)$ -mixing time for the source  $v$  and  $\tau(\epsilon)$ -mixing time of the graph)

Define  $\tau_v(\epsilon) = \min\{t : \|\mathbf{P}_t - \pi\|_1 \leq \epsilon\}$ , where  $\|\cdot\|_1$  is the  $L_1$  norm. Then  $\tau_v(\epsilon)$  is called the  $\epsilon$ -near mixing time for any  $\epsilon$  in  $(0, 1)$ . The mixing time of the graph is denoted by  $\tau(\epsilon)$

and is defined by  $\tau(\epsilon) = \max\{\tau_v(\epsilon) : v \in V\}$ . It is clear that  $\tau_v(\epsilon) \leq \tau(\epsilon)$ .

We note that mixing time definition usually takes  $\epsilon$  to be  $1/2e$ . Our goal is to estimate  $\tau_v(\epsilon)$  for any small  $\epsilon \in (0, 1)$ , say  $\epsilon = 1/n^2$ . We omit  $\epsilon$  from the notation when it is understood from the context. The definition of  $\tau_v$  is consistent due to the following standard monotonicity property of the random walk probability distributions. The proof can be found in [14] or from the Exercise 4.3 in [13].

LEMMA 1.2.  $\|\mathbf{P}_{t+1} - \pi\|_1 \leq \|\mathbf{P}_t - \pi\|_1$ .

## 1.3 Related Work

Das Sarma et al. [7] presented a fast decentralized algorithm for estimating mixing time, conductance and spectral gap of the network. In particular, they show that given a starting node  $s$ , the mixing time with respect to  $s$ , i.e.,  $\tau_s$ , can be estimated in  $\tilde{O}(n^{1/2} + n^{1/4}\sqrt{D\tau_s})$  rounds. This gives an alternative algorithm to the only previously known approach by Kempe and McSherry [10] that can be used to estimate  $\tau_s$  in  $\tilde{O}(\tau_s)$  rounds. In fact, the work of [10] does more and gives a decentralized algorithm for computing the top  $k$  eigenvectors of a weighted adjacency matrix that runs in  $O(\tau \log^2 n)$  rounds if two adjacent nodes are allowed to exchange  $O(k^3)$  messages per round, where  $\tau$  is the mixing time and  $n$  is the size of the network.

The algorithm of Das Sarma et al. [7] is based on sampling nodes by performing sub-linear time random walks of certain length and comparing the distribution with the stationary distribution. Their algorithm can be faster than our approach in certain cases, but slower in some cases than ours. In particular, if  $\tau$  is smaller than  $\max\{\sqrt{n}, n^{1/4}\sqrt{D}\}$ , then our algorithm is faster. Also there is a grey area for the accuracy parameter  $\epsilon$  for which their algorithm cannot estimate the mixing time. They use a testing result from Batu et al. [3] to determine if the two distributions are close enough. This test may fail if the difference between the two distributions falls in a certain interval. More precisely, the algorithm of Das Sarma et al. estimates the mixing time for accuracy parameter  $\epsilon = 1/(2e)$  with respect to a source node  $v$ ,  $\tau_v(1/2e)$  as follows: the estimated value will be between the true value and  $\tau_v(O(1/(\sqrt{n} \log n)))$ . In contrast, our algorithm estimates  $\tau_v(\epsilon)$ , for even small values of  $\epsilon$ , say  $\epsilon = 1/n^2$ : the estimated value will be between the true value and  $\tau_v(1/n^2)$ .

## 2. ALGORITHM FOR MIXING TIME

We present an algorithm to compute the mixing time of a graph  $G$  from a specified source node. In other words, we present an algorithm which finds a length  $\ell$  such that the probability distribution of a random walk of length  $\ell$  reaches close to the stationary distribution. The main idea of our algorithm is to perform many random walks from the source node of some length  $\ell$  in parallel. After  $\ell$  steps, every node  $u$  estimates the probability distribution  $\mathbf{P}_\ell(u)$  as the fraction of random walks that terminate at  $u$  over all the walks. Then we compare the estimated distribution of  $\mathbf{P}_\ell$  with the stationary distribution  $\pi$  to determine if they are sufficiently close; otherwise, we double the length  $\ell$  and retry. Once we find the correct consecutive lower and upper bound of the length, a binary search will determine the mixing length (up to the allowed accuracy parameter  $\epsilon$ ). The monotonicity property (cf. Lemma 1.2) admits the binary search.

Our algorithm starts with  $\ell = 1$  and runs  $K = 80n^8 \log n$  random walks of length  $\ell$  from a source node, say node  $s$ . When the difference (i.e., the  $L_1$ -norm difference) between the estimated  $\ell$ -length walk distribution of  $\mathbf{P}_\ell$  with the stationary distribution  $\pi$  is greater than  $\epsilon$ ,  $\ell$  is doubled and retried. This process is repeated to identify the largest  $\ell$  such that the difference is greater than  $\epsilon$  and the smallest  $\ell$  such that the difference is less than  $\epsilon$ . These give lower and upper bounds on the required  $\tau_s$  respectively. In fact, the upper bound (i.e., the smallest  $\ell$  such that the difference is less than  $\epsilon$ ) is at most twice as  $\tau_s$ , since we are doubling the length each time. Then a binary search between lower and upper bounds will determine the exact mixing time  $\tau_s$ . Note that the monotonicity property (cf. Lemma 1.2) guarantees that once the difference becomes less than  $\epsilon$ , then it would be always less for any larger length. We show in the analysis that the estimated probability of  $\mathbf{P}_\ell(u)$  is close to the actual probability for every node  $u$  in each step of the walk (cf. Lemma 2.1). The pseudocode is given in Algorithm 1.

We show (in the next section) that the above algorithm estimates mixing time accurately with high probability<sup>2</sup>. The main technical challenge in implementing the above method in CONGEST model is that performing many walks from a source node in parallel can create a lot of congestion. Our algorithm uses a crucial property of random walks to overcome the congestion (see e.g., [6, 7]). In particular, we show that there will be no congestion in the network even if we perform up to a polynomial (in  $n$ ) number of random walks from the source node in parallel (cf. Lemma 2.3). The basic idea is to send the *count* of the number of random walks that pass through an edge. As random walks are memoryless processes, it is sufficient to send the number of walks traversing an edge in a given round to estimate  $\mathbf{P}_\ell$ . Since this number is polynomially bounded,  $O(\log n)$  bits suffice. Therefore, it is easy to see that performing  $\ell$ -length of random walks finishes in  $O(\ell)$  rounds in CONGEST model. We show that our algorithm computes mixing time accurately in  $O(\tau_s \log n)$  rounds with high probability (cf. Theorem 2.4).

## 2.1 Reducing the number of random bits

To perform  $K = 80n^8 \log n$  random walks can require at least so many random bits per node (as well as that much time), if done in a straightforward manner. We can make the algorithm more lightweight by doing only  $O(d(u) \log n)$  work and only so many random coin tosses per node. Instead of doing coin-flips for each of the tokens separately, we do the following to reduce the number of random coin-flips overhead at each node. Nodes which have less than  $(\text{degree} \times \log n)$  tokens, will select an edge randomly for each token i.e., they will do the coin-flips for each token. At any round, if a node  $u$  has tokens  $T^u$  such that  $T^u \geq d(u) \log n$ , then  $u$  forwards the average  $T^u/d(u)$  tokens to all of its neighbors (there are  $d(u)$  many). We show that this simple averaging approach has the bias of at most  $O(\log n)$  with high probability. In fact, if we perform coin-flips for each token i.e., select a neighbor randomly for each token then, in expectation,  $T^u/d(u)$  tokens will choose to go to one neighbor, where  $T^u$  is the total number of tokens at  $u$ . Then by a Chernoff bound, it is easy to show that the deviation from the mean is at most  $O(\log n)$  with high probability. That is, the bias is  $O(\log n)$  tokens per edge with high probability. Therefore, the total bias at a node  $u$  is

<sup>2</sup>With high probability means with probability at least  $1 - \frac{1}{n}$ .

$O(d(u) \log n)$  in a single round ( $O(\log n)$  bias coming from  $d(u)$  neighbors). Hence, total bias at node  $u$  until the algorithm stops is  $O(\tau_s d(u) \log n)$ , since the algorithm runs for  $O(\tau_s)$  rounds. This quantity is at most  $n^4 \log n$ , since  $d(u) < n$  and  $\tau_s \leq O(n^3)$  in worst case. Therefore, the deterministic averaging method has the additional approximation error at most  $(n^4 \log n)/K = 1/n^4$ , which is negligible compared to the considered error  $\epsilon = 1/n^2$ .

---

### Algorithm 1 ESTIMATEMIXINGTIME

---

**Input:** A graph  $G = (V, E)$  and a source node  $s$ .

**Output:**  $\tau_s(1/n^2)$  (mixing time starting from the node  $s$ ).

- 1: Node  $s$  creates a BFS tree via flooding (rooted at  $s$ ) so that each node knows their parent in the BFS tree.
  - 2: Node  $s$  broadcasts the value  $K = 80n^8 \log n$  to all the other nodes via broadcasting over the tree.
  - 3: **for**  $h = 0, 1, 2, \dots$  **do**
  - 4:    $\ell \leftarrow 2^h$
  - 5:   Node  $s$  creates  $K = 80n^8 \log n$  random walk tokens.
  - 6:   **for** round  $i = 1, 2, \dots, \ell$  **do**
  - 7:     Each node  $v$  holding at least one token, does the following in parallel:
  - 8:       For every neighbor  $w$ , set  $T_w^v = 0$  // [ $T_w^v$  indicates the number of tokens chosen to move to  $w$  from  $v$  at round  $i$ ]
  - 9:       For all the tokens calculate the number  $T_w^v$  for each neighbor  $w$  (as described in Section 2.1).
  - 10:      Send the counter number  $T_w^v$  to the neighbor  $w$ .
  - 11:   **end for**
  - 12:   Each node  $w$  counts the total number of tokens it holds. Say,  $\zeta_w = \sum_{v \in N(w)} T_w^v$ .
  - 13:   Each node  $w$  sends the absolute difference value  $\partial_w = |\frac{\zeta_w}{K} - \frac{d(w)}{2m}|$  to node  $s$  through the BFS tree. Each node sums up the  $\partial_w$  values of their children and forwards the sum to its parent.
  - 14:   Node  $s$  locally checks:
  - 15:    **if**  $(\sum_{w \in V} \partial_w \leq 1/n^2)$  **then**
  - 16:      BREAK;
  - 17:    **else**
  - 18:      Continue from Step 4 for the next (doubling) value of  $\ell$ .
  - 19:   **end if**
  - 20: **end for**
  - 21: Let's say  $\ell = 2^l$  for some integer  $l$ .
  - 22: Perform binary search from  $2^{l-1}$  to  $2^l$  to find an integer  $\ell$  such that:
  - 23: For length  $\ell$ ,  $\sum_{w \in V} \partial_w \leq 1/n^2$ , and for length  $\ell - 1$ ,  $\sum_{w \in V} \partial_w > 1/n^2$ , by computing from Step 5 to Step 15.
  - 24: Output  $\ell$ .
- 

## 2.2 Analysis of the Algorithm

We first show a result which will imply that our algorithm correctly estimates the mixing time for a given source node. Let  $\mathbf{P}_\ell(u)$  be the probability that a random walk of length  $\ell$  terminates over node  $u$ , starting from the given source node. We show that the above process of performing many random walks and then computing the fraction of walks that stop at  $u$  can approximate  $\mathbf{P}_\ell(u)$  with high accuracy for every node  $u$ . For this, we can ignore the probability  $\mathbf{P}_\ell(u)$  whose value is less than  $1/n^4$ . Because in the worst case, there may be at most  $n$  such vertices and the sum of these ignored probabilities is less than  $1/n^3$ , which is negligible compared to  $1/n^2$ ,

the estimation error. (Here we assume that the mixing time estimation error  $\epsilon$  is  $1/n^2$  — see Definition 1.1). However, one can achieve much more accuracy by performing a larger number of random walks. In the following lemma, we show that if we perform  $K = 80n^8 \log n$  random walks from a source node  $s$ , then the above algorithm can estimate  $\mathbf{P}_\ell(u)$  at any node  $u$  for which  $\mathbf{P}_\ell(u) \geq 1/n^4$ .

**LEMMA 2.1.** *If the probability of an event  $X$  occurring is  $p$  such that  $p \geq 1/n^4$ , then in  $t = 80n^8 \log n$  trials, the fraction of times the event  $X$  occurs is  $p \pm \frac{1}{n^6}$  with high probability.*

**PROOF.** The proof follows from the standard Chernoff bound. We also included in the full version [14].  $\square$

**LEMMA 2.2.** *The algorithm ESTIMATETIME approximates the probability  $\mathbf{P}_\ell(u)$  for every length  $\ell$  and for all node  $u$  (such that  $\mathbf{P}_\ell(u) \geq 1/n^4$ ) with high probability.*

**PROOF.** Let's first consider a particular length  $\ell$ . Suppose the algorithm ESTIMATETIME outputs the estimated probability  $\tilde{\mathbf{P}}_\ell(u)$  for each node  $u$  for the length  $\ell$ . It follows from the Lemma 2.1 that  $\tilde{\mathbf{P}}_\ell(u) = \mathbf{P}_\ell(u) \pm 1/n^6$  with high probability for any node  $u$  for which  $\mathbf{P}_\ell(u) \geq 1/n^4$  (by taking union bound). Therefore,  $|\tilde{\mathbf{P}}_\ell(u) - \mathbf{P}_\ell(u)| \leq 1/n^6$  for any node  $u$  (such that  $\mathbf{P}_\ell(u) \geq 1/n^4$ ). Since  $\ell$  can be arbitrary, the lemma holds for every length  $\ell$ .  $\square$

Before proceeding to the main result (cf. Theorem 2.4) of this section, we prove a crucial lemma on the congestion of our algorithm. The lemma below guarantees that there will be no congestion even if we perform a polynomial number of random walks in parallel in the network. While the proof intuition is discussed before, the full proof is included in [14].

**LEMMA 2.3.** *If every node perform at most a polynomial (in  $n$ ) number of random walks in parallel then there will be no congestion in the network.*

**THEOREM 2.4.** *Given a graph  $G$  and a source node  $s$ , the algorithm ESTIMATETIME takes  $O(\tau_s \log n)$  rounds and finds the mixing time  $\tau_s(1/n^2)$  with high probability.*

**PROOF. Correctness.** The Lemma 2.2 says that when  $\ell$  reaches the mixing time of the graph, the estimated probability  $\tilde{\mathbf{P}}_\ell(u)$  of  $\mathbf{P}_\ell(u)$  will be close to the stationary distribution  $d(u)/2m$  for each node  $u$ . Hence, the  $L_1$ -norm difference is  $\sum_{u \in V} |\tilde{\mathbf{P}}_\ell(u) - d(u)/2m| \leq 1/n^5 + 1/n^3 < 1/n^2$ . The term  $1/n^3$  is coming from the fact that we neglected those probabilities such that  $\mathbf{P}_\ell(u) < 1/n^4$ ; so summing up all of them could be at most  $1/n^3$ . Further, it follows from the monotonicity property (cf. Lemma 1.2) that once the norm difference becomes less than  $1/n^2$ , it would always be less than  $1/n^2$  for any larger length of the random walk. Hence, our algorithm terminates when the mixing time is correctly estimated (cf. Definition 1.1).

**Running time.** To estimate the mixing time, we try out increasing values of  $\ell$  that are powers of 2. Once we find the right consecutive powers of 2, the monotonicity property admits a binary search to determine the exact value. Since each node  $u$  knows its own stationary probability (determined just by its degree), they can compute the difference  $\partial_u = |\frac{\zeta_u}{K} - \frac{d(u)}{2m}|$  locally and send it to the source node  $s$ . Recall that  $\zeta_u$  is the number of walks that stop at  $u$  after  $\ell$  steps

and  $K$  is the total number of random walks started initially. The node  $s$  eventually collects the sum of all  $\partial_u$ s and checks if  $\sum_{u \in V} \partial_u$  is less than  $1/n^2$ . Performing  $K$  random walks of length  $\ell$  can be done in  $O(\ell)$  rounds, as there is no congestion (cf. Lemma 2.3). Further, sending the sum of all  $\partial_u$ s to the source node  $s$ , can be done in  $O(D)$  rounds ( $D$  is the diameter). Since this can be done by an upcast through the BFS tree rooted at  $s$  (see Step 13 of Algorithm 1). Hence, for a particular  $\ell$ , it requires  $O(\ell + D)$  rounds. Therefore, total time required to compute the mixing time for a source node  $s$  is  $O(\sum_{i=1}^{\log \tau_s} (\ell + D))$  rounds which is  $O((\tau_s + D) \log n)$  rounds as  $\ell$  is at most  $\tau_s$  which is polynomially bounded. Notice that when  $\ell \geq \tau_s(1/n^2)$ , then the  $L_1$  difference between estimated distribution and stationary distribution becomes less than  $1/n^2$  (correctness follows from the Lemma 2.2). This gives the estimated mixing length at most twice as  $\tau_s(1/n^2)$ , since we are doubling the length each time. Then a binary search will determine the exact mixing time  $\tau_s(1/n^2)$ . The binary search takes at most  $O((\tau_s + D) \log n)$  rounds. Therefore, the algorithm finishes in  $O(\tau_s \log n)$  rounds, since  $D \leq \Omega(\tau_s)$  for any graph.  $\square$

### 3. REFERENCES

- [1] J. Augustine, A. R. Molla, E. Morsy, G. Pandurangan, P. Robinson, and E. Upfal. Search and storage in dynamic peer-to-peer networks. In *SPAA*, pages 53–62, 2013.
- [2] J. Augustine, G. Pandurangan, and P. Robinson. Fast byzantine agreement in dynamic networks. In *PODC*, pages 74–83, 2013.
- [3] T. Batu, E. Fischer, L. Fortnow, R. Kumar, R. Rubinfeld, and P. White. Testing random variables for independence and identity. In *FOCS*, pages 442–451, 2001.
- [4] A. Das Sarma, S. Gollapudi, and R. Panigrahy. Sparse cut projections in graph streams. In *ESA*, pages 480–491, 2009.
- [5] A. Das Sarma, A. R. Molla, and G. Pandurangan. Distributed computation of sparse cuts via random walks. In *ICDCN*, pages 6:1–6:10, 2015.
- [6] A. Das Sarma, A. R. Molla, G. Pandurangan, and E. Upfal. Fast distributed pagerank computation. *Theor. Comput. Sci.*, 561:113–121, 2015.
- [7] A. Das Sarma, D. Nanongkai, G. Pandurangan, and P. Tetali. Distributed random walks. *J. ACM*, 60(1):2, 2013.
- [8] C. Gkantsidis, M. Mihail, and A. Saberi. Throughput and congestion in power-law graphs. In *SIGMETRICS*, pages 148–159, 2003.
- [9] M. Jerrum and A. Sinclair. Approximating the permanent. *SICOMP*, 18(6):1149–1178, 1989.
- [10] D. Kempe and F. McSherry. A decentralized algorithm for spectral analysis. *JCSS*, 74(1):70–83, 2008.
- [11] F. Kuhn and A. R. Molla. Distributed sparse cut approximation. In *OPODIS 2015*, volume 46, pages 1–14, 2016.
- [12] S. Kutten, G. Pandurangan, D. Peleg, P. Robinson, and A. Trehan. On the complexity of universal leader election. *J. ACM*, 62(1):7:1–7:27, 2015.
- [13] D. A. Levin, Y. Peres, and E. L. Wilmer. *Markov Chains and Mixing times*. AMS, USA, 2008.
- [14] A. R. Molla and G. Pandurangan. Distributed computation of mixing time. In *arXiv:1610.05646*, 2016.
- [15] G. Pandurangan. Distributed algorithmic foundations of dynamic networks. In *SIROCCO*, pages 18–22, 2014.
- [16] G. Pandurangan and M. Khan. Theory of communication networks. In *Algorithms and Theory of Computation Handbook*. CRC Press, 2009.
- [17] D. Peleg. *Distributed computing: a locality-sensitive approach*. SIAM, Philadelphia, PA, USA, 2000.
- [18] D. A. Spielman and S. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC*, pages 81–90, 2004.

# Fault-tolerant Gathering of Semi-synchronous Robots

Subhash Bhagat

Advanced Computing and Microelectronics Unit  
 Indian Statistical Institute  
 203 B.T. Road  
 Kolkata, India  
 sbhagat\_r@isical.ac.in

Krishnendu Mukhopadhyaya

Advanced Computing and Microelectronics Unit  
 Indian Statistical Institute  
 203 B. T. Road  
 Kolkata, India  
 krishnendu@isical.ac.in

## ABSTRACT

This paper addresses the *Gathering* problem which asks robots to gather at a single point which is not fixed in advance, for a set of small, autonomous, mobile robots. The problem is studied for a set of semi-synchronous robots under *SSYNC* model when the robots may become faulty (crash fault). Depending upon the capabilities of the robots, the algorithms are designed to tolerate maximum number of faults. This work assumes weak multiplicity detection capability of the robots. The contribution of this work is in two folds. First, a distributed algorithm is presented which can tolerate at most ( $\lfloor \frac{n}{2} \rfloor - 1$ ) crash faults for  $n \geq 7$  robots with weak multiplicity detection only. For the second algorithm, it is also assumed that robots know the mobility capacity of all the robots. The algorithm presented here can tolerate at most  $(n - 6)$  crash faults for  $n \geq 7$  robots.

## CCS Concepts

- Theory of computation → Design and analysis of algorithms; Distributed algorithms; Self-organization;

## Keywords

Swarm robotics, semi-synchronous robots, oblivious, gathering, crash faults

## 1. INTRODUCTION

The collective and cooperative behaviours of a set of small, autonomous, inexpensive mobile robots have been the main focus of *Swarm robotics*. The robots are autonomous i.e., they work without any centralized control. In general settings, they are oblivious, homogeneous and anonymous. Since they are oblivious, they do not carry forward any information of their previous computational cycles. Homogeneity means that all the robots have same capabilities and they execute same algorithm. The anonymity of the robots makes them indistinguishable by their nature or identity.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

*ICDCN '17, January 04-07, 2017, Hyderabad, India*

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007781>

The robots do not have any explicit communication abilities. However, implicit communications are achieved via sensing the locations of the robots in the system. They lack of any global coordinate system. Each robot has its own local coordinate system which may differ from others in orientation, directions of axes and unit distance.

Each robot follows same *Look-Compute-Move* cycle repeatedly. First, an active robot takes the snapshot of its surroundings to obtain the locations of the robots w.r.t. its local coordinate system (*Look* phase). This information is used to compute a destination point (*Compute* phase). Finally, the robot moves towards this computed destination point (*Move* phase). The activations of the robots, the timings of the operations and the completion times depend on the scheduler. An asynchronous scheduler (*ASYNC*) does not impose any restriction on the activation of the robots [14]. The timing of the operations and their completion times are unpredictable but finite. A semi-synchronous (*SSYNC*) scheduler discretizes the time into several rounds [16]. In each round, it allows a subset of robots to be activated simultaneously and to operate all together instantaneously. The unpredictability lies in the activated subset of robots in each round. A *fully synchronous* scheduler (*FSYNC*) is the strongest of these schedulers which allows all robots to be activated in all rounds. We assume a fair scheduler which activates each robot infinitely often.

The robots may have some additional capabilities. The *weak multiplicity detection* capability enables a robot to identify the multiple occurrences of robots at a single point. Whereas, *strong multiplicity detection* helps them to count the total number of robots at that location. The robots may have common *chirality* (i.e., clockwise direction). They can also have agreements on the directions and orientations of the axes of their local coordinate systems. The mobility of a robot may be *rigid* or *non-rigid*. In *rigid* motion, a robot reaches its destination without halting in between. In *non-rigid* movements, a robot may be stopped by an adversary before reaching its destination point. However, to guarantee finite time reachability for the robots, it is assumed that a robot always moves at least a distance  $\min\{\delta, d\}$  towards its destination point where  $d$  is the distance of its destination point from its current position, for some constant  $\delta > 0$ .

Another aspect of the system comes from the fact that the robots may become faulty at any stage of execution. Three basic types of faults are considered. The *transient* fault corrupts the memory of the robots. The obliviousness of the robots, makes them naturally resilient to this kind

of faults. The *crash* fault impairs the robots to perform any kind of action. However, they physically remains in the system. The *byzantine* fault is a malicious type of fault. A byzantine robot behaves arbitrarily. The algorithms which tolerate any of these faults, should successfully terminate for correct robots in finite time. By  $(n, f)$ , we denote the model which permits at most  $f$  faulty robots among total  $n$  robots.

This paper considers the *gathering* problem under crash fault model. The problem is defined as follows: a set of robots is deployed in the two dimensional Euclidean plane. The non-faulty robots should coordinate their movements to meet at a single point, not fixed a priori, in finite time.

## 1.1 Earlier works

The gathering problem has been studied extensively under different models and considering different capabilities of the robots [11]. The primary goal is to identify minimal sets of constraints needed to solve the problem. In FSYNC model, gathering is solvable even with  $f < \frac{n}{3}$  byzantine robots [2]. Under SSYNC model, gathering is deterministically unsolvable in the absence of multiplicity detection and any form of coordinate axes agreement [15]. One of the most significant works in the crash fault model for semi-synchronous robots is presented by Bramas and Tixeuil [6]. The work presented a distributed algorithm for the gathering problem under  $(n, n - 1)$  crash fault model for semi-synchronous robots with strong multiplicity detection capability. Izumi et al. [13] proved that the problem is deterministically unsolvable in the presence of a single byzantine robot even if robots have agreement in both coordinate axes, with unlimited mobility and they are not oblivious. Defago et al. [10] presented a study of probabilistic gathering under crash faults and byzantine faults considering different types of schedulers. In ASYNC model, gathering is deterministically solvable for  $n > 2$  robots with weak multiplicity detection [8]. When robots have limited visibility range, it is shown that gathering is possible if robots have agreements in direction and orientation of the both axes [12]. Bhagat et al. [3] proved that gathering is solvable for asynchronous robots in the presence of arbitrary number of crash faults under one axis agreement even if robots are opaque i.e., they obstruct the visibility of the other robots. Gathering problem has also been studied for *fat* robots (robots are represented as unit discs) [1, 4, 9, 7].

## 1.2 Our Contribution

We consider the gathering problem with weak multiplicity detection under *SSYNC* model when robots may develop crash faults. In these settings, Agmon and Peleg [2] first considered the problem and presented a distributed algorithm to solve the problem which can tolerate a single crash fault. This paper proposes algorithms which solve the problem with more number of admissible crash faults. The contribution of this paper is in two folds. First, it proposes a distributed algorithm to solve the gathering problem for a set of  $n \geq 7$  semi-synchronous robots in  $(n, \lfloor \frac{n}{2} \rfloor - 1)$  crash fault model. Secondly, the problem is solved for  $(n, n - 6)$  crash fault robots when robots have the knowledge of  $\delta$  also.

Following is the organization of the paper: Section 2 states the assumptions of the robot model used in this paper and presents the definitions and notations used to describe the algorithms. Sections 3 and 4 explain the two algorithms

for the gathering problem and the corresponding proofs of correctness are given. Finally the section 5 presents the conclusion.

## 2. GENERAL MODEL AND DEFINITIONS

The system consists of  $n$  homogeneous, autonomous, oblivious robots. The robots are represented as points in the two-dimensional Euclidean plane where they can move freely. They do not share any global coordinate system. However, each robot has its own local coordinate system centred at its current position. The directions and orientations of the axes and the unit distance may vary from other robots. They do not share any common chirality. They also lack any form of explicit communication capability. We assume that initially all the robots occupy distinct positions. Each robot has unlimited visibility range. We consider the *SSYNC* model with some additional assumptions. The movements of the robots are *non-rigid* i.e., a robot moves at least a distance  $\delta$  towards its destination point, if it does not reach its destination. A robot may develop crash fault at any stage of execution i.e., the model is the  $(n, f)$  crash fault model. The robots are endowed with weak multiplicity detection capability.

- **Configuration of the robots:** The set of  $n$  robots is denoted by  $\mathcal{R} = \{r_1, r_2, \dots, r_n\}$ . Let  $r_i(t)$  be the point occupied by the robot  $r_i$  at time  $t$ . A robot configuration is denoted by the multi set  $\mathcal{R}(t) = \{r_1(t), \dots, r_n(t)\}$ . Let  $\tilde{\mathcal{R}}$  denote the set of all configurations which contain at most one multiplicity point (a point containing more than one robot on it).
- **Measurement of angles:** Since the robots do not have common chirality, the angle between two given line segments is considered as the angle which is less than or equal to  $\pi$ .
- **Smallest Enclosing Circle:** Let us denote the smallest enclosing circle of the points in  $\mathcal{R}(t)$  by  $SEC(\mathcal{R}(t))$  and its centre by  $\mathcal{O}_t$ . We define two sets  $C_{out}(t)$  and  $C_{int}(t)$  as the collections of robot positions on the circumference of  $SEC(\mathcal{R}(t))$  and the robot positions lying strictly within  $SEC(\mathcal{R}(t))$  respectively. For each robot  $r_i \in \mathcal{R}$  such that  $r_i(t) \neq \mathcal{O}_t$ , let  $rad_i(t)$  denote the half line starting from  $\mathcal{O}_t$  (but excluding  $\mathcal{O}_t$ ) and passing through  $r_i(t)$  (Figure 1(a)). Let  $|rad_i(t)|$  denote the number of distinct robot positions on  $rad_i(t)$ . Note that  $1 \leq |rad_i(t)| \leq n - 1$ . When there is no ambiguity, we use  $SEC(t)$  instead of  $SEC(\mathcal{R}(t))$ .
- Let  $(a, b)$  and  $\overline{ab}$  denote the open and closed line segments joining the points  $a$  and  $b$  respectively (excluding and including the two end points  $a$  and  $b$  respectively). Let  $|a, b|$  denote the distance between the points  $a$  and  $b$ . For two sets  $A$  and  $B$ , by  $A \setminus B$ , we denote the set difference of  $A$  and  $B$ .

We use some concepts defined in [5]. Following is the list of these definitions and notions:

- **View of a robot:** The view  $\mathcal{V}(r_i(t))$  of a robot  $r_i \in \mathcal{R}$  is defined as the set of polar coordinates of the points in

$\mathcal{R}(t)$ , where the polar coordinate system of  $r_i$  is defined as follows: (i) the point  $r_i(t)$  is the origin of the coordinate system and (ii) the point  $(1, 0)$  is  $\mathcal{O}_t$  if  $r_i(t) \neq \mathcal{O}_t$ , otherwise it is any point  $r_k(t) \neq r_i(t) \in \mathcal{R}(t)$  that maximizes  $\mathcal{V}(r_k(t))$ . The orientation of the polar coordinate system should maximize  $\mathcal{V}(r_i(t))$ . The view of each robot is defined uniquely. The views of two robots are compared in lexicographic order.

- **Rotational symmetry:** Let a relation  $\sim$  be defined on  $\mathcal{R}(t)$  as follows:  $\forall r_i(t), r_j(t) \in \mathcal{R}(t), r_i(t) \sim r_j(t)$  if and only if  $\mathcal{V}(r_i(t)) = \mathcal{V}(r_j(t))$  with same orientation. The relation  $\sim$  is an equivalence relation and it partitions  $\mathcal{R}(t)$  into disjoint equivalence classes. Let  $\text{sym}(\mathcal{R}(t))$  denote the cardinality of the largest equivalence class defined by  $\sim$ . The set  $\mathcal{R}(t)$  is said to be rotationally symmetric if  $\text{sym}(\mathcal{R}(t)) > 1$  (Figure 1(b)).

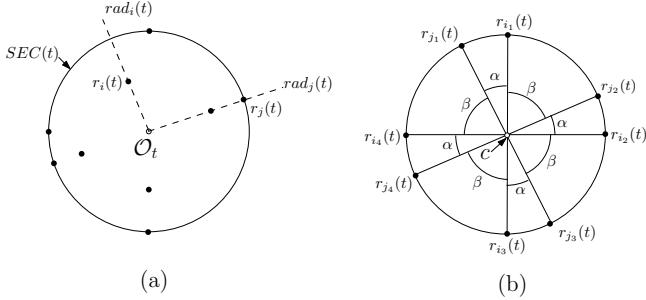


Figure 1: An example of (a) an illustration of  $\text{SEC}(t)$ ,  $\text{rad}_i(t)$ ,  $\mathcal{O}_t$  and  $|rad_j(t)| = 2$  (b) a symmetric configuration with  $\text{sym}(C) = 4$  where  $\{r_{i1}, r_{i2}, r_{i3}, r_{i4}\}$  and  $\{r_{j1}, r_{j2}, r_{j3}, r_{j4}\}$  are two equivalence classes

- **Successor:** Let us consider a robot configuration  $\mathcal{R}(t)$  and a fixed point  $c \in \mathbb{R}^2$ . Suppose,  $S(r_i(t), c)$  denotes the clockwise successor of a point  $r_i(t) \in \mathcal{R}(t)$  around  $c$  which is defined as follows: if  $\mathcal{R}(t) \cap (c, r_i(t)) \neq \phi$ , then the point  $r_j(t) \in \mathcal{R}(t) \cap (c, r_i(t))$  which minimizes  $|r_i(t), r_j(t)|$ , is the clockwise successor of  $r_i$ . Otherwise,  $r_j(t)$  is the point in clockwise direction such that  $\angle(r_i(t), c, r_j(t))$  contains no other point of  $\mathcal{R}(t)$  and  $|c, r_j(t)|$  is maximized. The  $k^{th}$  clockwise successor of  $r_i(t)$  around  $c$ , denoted by  $S^k(r_i(t), c)$ , is defined by the recursive relation: for  $k > 1$ ,

$$S^k(r_i(t), c) = S(S^{k-1}(r_i(t), c), c), \text{ where } S^1(r_i(t), c) = S(r_i(t), c) \text{ and } S^0(r_i(t), c) = r_i(t).$$

The counter-clockwise successor of  $r_i$  can be defined analogously.

- **String of angles:** Let  $\text{mult}(c)$  denote the number of robots occupying the point  $c$ . Let  $SA(r_i(t), c)$  denote the string of angles  $\alpha_1(t), \alpha_2(t), \dots, \alpha_m(t)$  where  $m = n - \text{mult}(c)$  and  $\alpha_i(t) = \angle(S^{i-1}(r_i(t)), c, S^i(r_i(t)))$ . The length of  $SA(r_i(t), c)$  is denoted by  $|SA(r_i(t), c)| = m$ . The string  $SA(r_i(t), c)$  is  $k$ -periodic if there exists a constant  $1 \leq k \leq m$  such that  $SA(r_i(t), c) = X^k$ , where  $X$  is a sub-string of  $SA(r_i(t), c)$ . The periodicity of  $SA(r_i(t), c)$ , denoted by  $\text{per}(SA(r_i(t), c))$ , is the largest value of  $k$  for which  $SA(r_i(t), c)$  is  $k$ -periodic (Figure 2(a)).

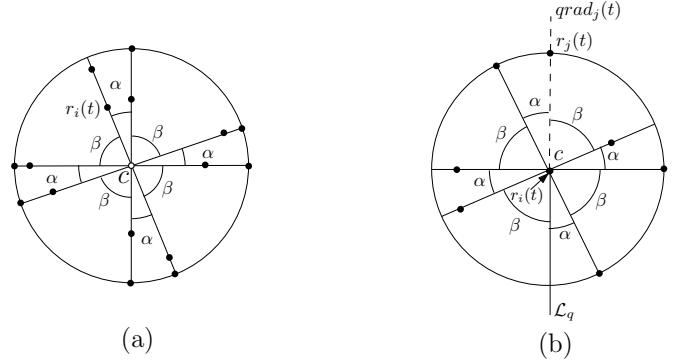


Figure 2: An example of (a) a regular configuration with  $\text{reg}(C) = 4$  where  $SA(r_i(t), c) = (0, \alpha, 0, \beta)^4$  (b) a Q-regular configuration with  $qreg(C) = 4$  where the robot at  $c$  is  $r_i$  and if it is shifted to any point on  $L_q$  the configuration becomes regular

- **Regularity:** A robot configuration  $\mathcal{R}(t)$  is said to be *regular* if  $\exists$  a point  $c \in \mathbb{R}^2$  and an integer  $m$  such that  $\text{per}(SA(r_i(t), c)) = m > 1 \forall r_i(t) \in \mathcal{R}(t)$ . The regularity of  $\mathcal{R}(t)$  is denoted by  $\text{reg}(\mathcal{R}(t)) = m$ . The point  $c$  is called the centre of regularity (Figure 2(a)).

- **Quasi Regularity:** We consider the quasi regularity only for those configurations in  $\tilde{\mathcal{R}}$  which contains no multiplicity point. A robot configuration  $\mathcal{R}(t) \in \tilde{\mathcal{R}}$  is said to be *quasi regular* or *Q-regular* if and only if  $\exists$  a configuration  $\mathcal{B}(t)$  and a point  $c \in \mathbb{R}^2$  such that  $\text{reg}(\mathcal{B}(t)) > 1$ ,  $c$  is the centre of regularity of  $\mathcal{B}(t)$  and  $p \in \mathcal{R}(t) \setminus \mathcal{B}(t)$  implies that  $p = c$ . In other words,  $\mathcal{B}(t)$  can be obtained from  $\mathcal{R}(t)$  only by moving the robot position at  $c$ , if any, along a particular half line starting at  $c$  (including  $c$ ). Let us denote this half line by  $L_q$ . If  $\mathcal{R}(t)$  is Q-regular, then the point  $c$  is called the centre of Q-regularity. By  $c_q$ , we denote the centre of Q-regularity. The quasi-regularity of  $\mathcal{R}(t)$  is denoted by  $qreg(\mathcal{R}(t)) = \text{reg}(\mathcal{B}(t))$ . If  $\mathcal{R}(t)$  is not quasi-regular, then  $qreg(\mathcal{R}(t)) = 1$ . For each robot position  $r_i(t) \in \mathcal{R}(t)$ , let  $qrad_i(t)$  denote the half line starting from  $c$  (but excluding  $c$ ) and passing through  $r_i(t)$  if  $r_i(t) \neq c$ . Otherwise,  $qrad_i(t)$  is the line segment  $L_q$  (Figure 2(b)).

If  $\mathcal{R}(t)$  is Q-regular, we define an equivalence relation  $\sim_q$  on  $\mathcal{R}(t)$ :  $\forall r_i(t), r_j(t) \in \mathcal{R}(t), r_i(t) \sim_q r_j(t)$  iff  $qrad_i(t) = qrad_j(t)$ . Let  $[qrad_i(t)]$  denote the equivalence class in which  $r_i(t)$  belongs to and  $||qrad_i(t)||$  denote the number of distinct robot positions in this class. Let  $e_q(t)$  denote the total number of different equivalence classes defined by  $\sim_q$  on  $\mathcal{R}(t)$ . For a non-linear configuration, the centre of Q-regularity coincides with its unique Weber point and it is computable in finite time [5]. Note that if a configuration has multiple lines of symmetry, then it is Q-regular. However, the converse is not true.

### 3. ALGORITHM WITH UNKNOWN $\delta$

This section describes the study of the gathering problem under crash fault model when robots do not have the knowledge of  $\delta$ . To guarantee a finite time gathering in a fault

prone system, we have to show that the non-faulty robots in the system find their ways to gather at a point in finite time. While designing the algorithm, the things to be taken into considerations are: (i) since robots have weak multiplicity detection capability, a robot should not collide during the motion to avoid creation of more than one multiplicity point and (ii) there should not be any dead-lock or live-lock in the system. The main idea of the algorithm is to create a unique point of multiplicity where correct robots will gather finally. Different strategies are adopted to create a unique point of multiplicity. Our objective is to design a distributed algorithm which can tolerate at most  $\lfloor \frac{n}{2} \rfloor - 1$  crash faults. If in the initial configuration  $\mathcal{R}(t_0)$ , the set  $C_{int}(t_0)$  contains at least two non-faulty robots, the algorithm instructs all of them to move towards  $\mathcal{O}_{t_0}$  to create a unique point of multiplicity, avoiding collisions during movements. Otherwise, our strategy is to move robots from the set  $C_{out}(t_0)$  within the circle  $SEC(t_0)$  to ensure that  $C_{int}(t)$  contains at least two correct robots, for some  $t > t_0$ . Depending upon the initial configuration different scenarios arise.

### 3.1 Different Configurations

We divide  $\tilde{\mathcal{R}}$  into the following sub-classes:

- **Multiple ( $\mathcal{M}$ ):** It contains all those configurations which has a unique point of multiplicity.
- **Dense ( $\mathcal{D}$ ):** All the configurations for which  $|C_{int}(t)| \geq \lfloor \frac{n}{2} \rfloor + 1$ , belongs to this class.
- **Non-Dense ( $\mathcal{ND}$ ):** It contains all the configurations for which  $|C_{int}(t)| < \lfloor \frac{n}{2} \rfloor + 1$ .

It is easy to see that  $\tilde{\mathcal{R}} = \mathcal{M} \cup \mathcal{D} \cup \mathcal{ND}$ .

### 3.2 Algorithm GatheringFault()

We assume that in the initial configuration  $\mathcal{R}(t_0)$ , all the robots occupy distinct positions. An active robot  $r_i \in \mathcal{R}$  first checks in which sub-class  $\mathcal{R}(t)$  belongs to and accordingly takes the decision. If  $r_i$  finds a unique multiplicity point  $p_m$  i.e.,  $\mathcal{R}(t) \in \mathcal{M}$ , then it does one of the followings: (i) if  $r_i(t) \neq p_m$ , it moves towards  $p_m$  using the algorithm  $MoveToDestination()$  or (ii) otherwise, it does not move. If there is no such point  $p_m$ , the robot  $r_i$  executes algorithm  $CreateMultiplicity()$  to create a unique multiplicity point. The robots use  $MoveToDestination()$  to reach their respective destination points. Algorithm  $MoveToDestination()$  takes care of collision-less movements for the robots so that during the whole execution of the gathering algorithm no more than one point of multiplicity can be created.

---

#### ALGORITHM 1: GatheringFault()

---

```

Input:  $r_i \in R$ 
Output:  $r_i$  moves towards its destination.
if  $\mathcal{R}(t) \in \mathcal{M}$  then
|  $r \leftarrow p_m$ ;
else
|  $r \leftarrow CreateMultiplicity(\mathcal{R}(t))$ ;
end
Move to  $r$  using  $MoveToDestination(\mathcal{R}(t), r)$  ;

```

---

### 3.3 Algorithm MoveToDestination()

In this section, we describe an algorithm which robots use to reach their respective destination points without colliding with other robots. Since robots have weak multiplicity detection capability, the robots should avoid creation of multiple points of multiplicity. Followings are the different scenarios:

**Case-1 Robots have specific destinations:** First consider the case when a robot  $r_i \in \mathcal{R}$  has a specific destination point. Let  $w$  be a destination point of the robot  $r_i$  at time  $t$ . If current position of  $r_i$  coincides with the destination point i.e.,  $r_i(t) = w$ , the robot  $r_i$  does not move. Otherwise, the robot  $r_i$  uses the following strategies to reach  $w$ :

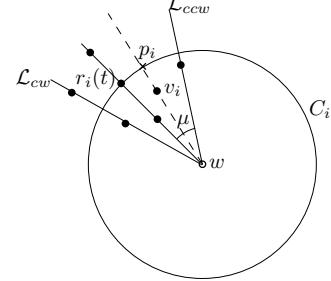


Figure 3: An illustration of scenario (a) in the algorithm  $MoveToDestination()$

- If  $(r_i(t), w) \cap \mathcal{R}(t) = \phi$  i.e., there is no other robot position in between  $r_i(t)$  and  $w$  on the line segment  $r_i(t)w$ , then  $r_i$  moves straight towards  $w$  along  $r_i(t)w$ .
- Otherwise,  $r_i$  does the followings:

(a) Consider all distinct half lines starting from  $w$  (excluding  $w$ ) such that each of them contains at least one robot position from  $\mathcal{R}(t)$ . Let  $\mathcal{L}_i$  be the half line that passes through  $r_i(t)$ . Let  $\mathcal{L}_{cw}$  and  $\mathcal{L}_{ccw}$  be the two closest half lines from  $\mathcal{L}$  in clockwise and counter-clockwise directions (according to the local coordinate system of  $r_i$ ) respectively. Let  $\mu_{cw}$  and  $\mu_{ccw}$  be the two angles made by  $\mathcal{L}_{cw}$  and  $\mathcal{L}_{ccw}$  with  $\mathcal{L}_i$  at the point  $w$ . Let  $\mu = \max\{\mu_{cw}, \mu_{ccw}\}$ . Let  $\mathcal{D}$  be the wedge defined by the angle  $\mu$  (Figure 3).

(b) Let  $v_i$  be the point in the wedge  $\mathcal{D}$  such that  $\angle(r_i(t), w, v_i) = \frac{1}{3ls} \mu$ , where  $l$  is total number of robots on  $r_i(t)w$  (excluding the robots at  $w$ ) and  $s$  is total number of robots between  $r_i(t)$  and  $w$  on  $r_i(t)w$ . Consider the circle  $C_i$  centred at  $w$  and passing through  $r_i(t)$ . Let  $C_i$  intersect  $wv_i$  at  $p_i$ . The robot  $r_i$  moves straight towards  $p_i$ .

**Case-2 Robots want to move inside  $SEC(t)$ :** Consider the case when a robot having position in  $C_{out}(t)$ , wants move inside  $SEC(t)$ . Let  $r_i \in \mathcal{R}$  be a robot which wants to move inside  $SEC(t)$ . If  $rad_i(t)$  contains only  $r_i(t)$ , then the destination point of  $r_i$  is  $\mathcal{O}_t$  and it moves straight towards this point. Otherwise, let  $r_j(t)$  be the closest robot position on  $rad_i(t)$  from  $r_i(t)$ . The middle point of  $r_i(t)r_j(t)$  is the destination point for  $r_i$  and it moves towards this point.

### 3.4 Algorithm *CreateMultiplicity()*

For a robot configuration  $\mathcal{R}(t)$ , different solution approaches are adopted depending upon the class in which  $\mathcal{R}(t)$  belongs.

- **Case-1:**  $\mathcal{R}(t) \in \mathcal{D}$  ( $|C_{int}(t)| \geq \lfloor \frac{n}{2} \rfloor + 1$ )  
Each active robot in  $C_{int}(t)$  moves towards  $\mathcal{O}_t$ , following algorithm *MoveToDestination()*. The robots in  $C_{out}(t)$  do not move.
- **Case-2:**  $\mathcal{R}(t) \in \mathcal{ND}$  ( $|C_{int}(t)| < \lfloor \frac{n}{2} \rfloor + 1$ )

- **Case-2.1:**  $\mathcal{R}(t) \notin \mathcal{QR}$  and  $\mathcal{R}(t)$  has no line of symmetry

In this case,  $|C_{out}(t)| \geq 4$  (since  $n \geq 7$ ). The robot positions in  $\mathcal{R}(t)$  are orderable [7]. Let  $\mathcal{H}(t)$  be an ordered set of the robot positions in  $\mathcal{R}(t)$ . Consider the robot  $r_i(t) \in C_{out}(t)$  which has highest order in  $\mathcal{H}(t)$ . Let  $\mathcal{L}_i(t)$  be the straight line joining  $r_i(t)$  and  $\mathcal{O}_t$ . Let  $v \neq r_i(t)$  be the other point of intersection between  $\mathcal{L}_i(t)$  and  $SEC(t)$ . Let  $r_l(t)$  and  $r_k(t)$  be the two robot positions in  $C_{out}(t)$  such that they lie on two different sides of  $\mathcal{L}_i(t)$  and they are closest to  $v$ . Let  $\mathcal{F}_1(t) = \{r_i(t), r_l(t), r_k(t)\}$ . The robots in  $\mathcal{F}_1(t)$  do not move. Rest of the robots in  $C_{out}(t)$  move inside  $SEC(t)$  using strategy stated in case-2 of algorithm *MoveToDestination()*. This will satisfy the condition  $|C_{int}(t')| \geq \lfloor \frac{n}{2} \rfloor + 1$ ,  $t' > t$ . A robot  $r_s$  in  $C_{int}(t)$  moves towards  $\mathcal{O}_t$ .

- **Case-2.2:**  $\mathcal{R}(t) \notin \mathcal{QR}$  and  $\mathcal{R}(t)$  has exactly one line of symmetry

Let  $\mathcal{L}$  be the line of symmetry. There are two scenarios:

- \* **Case-2.2.1:**  $\mathcal{L}$  passes through at least one point in  $C_{out}(t)$

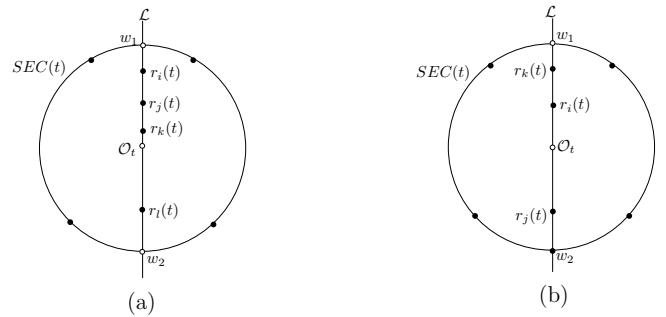
First, suppose,  $\mathcal{L}$  passes through exactly one point, say  $r_i(t)$ , in  $C_{out}(t)$ . Fix  $r_i(t), r_l(t)$  and  $r_k(t)$  where  $r_l(t)$  and  $r_k(t)$  are found by same technique as in the asymmetric case above using  $\mathcal{L}$  instead of  $\mathcal{L}_i(t)$ . Let  $\mathcal{F}_3(t) = \{r_i(t), r_l(t), r_k(t)\}$ . Now, consider the case when  $\mathcal{L}$  passes through two robot positions, say  $r_c(t)$  and  $r_d(t)$  in  $C_{out}(t)$ . Let  $\mathcal{F}_2(t) = \{r_c(t), r_d(t)\}$ . The robots having positions in  $\mathcal{F}_3(t)$  and  $\mathcal{F}_2(t)$  do not move. Rest of the robots in  $C_{out}(t)$  move inside  $SEC(t)$  and each robot  $r_i$  in  $C_{int}(t)$  moves towards  $\mathcal{O}_t$ . They use algorithm *MoveToDestination()* to reach their destination.

- \* **Case-2.2.2:**  $\mathcal{L}$  does not pass through any point in  $C_{out}(t)$

There are four points on  $C_{out}(t)$  which are closest to  $\mathcal{L}$ . Let  $\mathcal{F}_4(t)$  denote the set of these robot positions. The robots at these four points do not move. Rest of the robots on  $C_{out}(t)$  move inside  $SEC(t)$ . A robot  $r_i$  in  $C_{int}(t)$  moves towards  $\mathcal{O}_t$ , except the following two special case:

(a) For  $n = 8$ ,  $|C_{int}(t)| = 4$  and all the robots in  $C_{int}(t)$  lie on  $\mathcal{L}$ . Since there could be 3 faulty robots in  $C_{int}(t)$ , only a single robot could be able to reach  $\mathcal{O}_t$  and the configuration may remain symmetric. This could lead to a dead-lock in the system. To handle this, the robots do the following: Let  $\mathcal{L}$  intersects  $SEC(t)$  at the point  $w_1$  and  $w_2$  (Figure 4(a)). If the line segments  $\overline{\mathcal{O}_t w_1}$  and  $\overline{\mathcal{O}_t w_2}$  contain same number of robot positions, the nearest robots from  $\mathcal{O}_t$  on these two line segments, move towards  $\mathcal{O}_t$  and other two robots move towards  $w_1$  and  $w_2$  respectively. Otherwise, one of the lines, say  $\overline{\mathcal{O}_t w_1}$ , contains more robot positions than the other one. The robots on these two line segments having free corridors to  $w_1$  and  $w_2$ , move towards these points. The other robots in between them move towards the next robot position on  $\overline{\mathcal{O}_t w_1}$  and in the direction of  $w_1$ . Let  $d_i$  be the destination point of a robot  $r_i \in C_{int}(t)$ . Note that  $\mathcal{R}(t)$  remains symmetric during movements of the robots on  $\mathcal{L}$ .

(b) For  $n = 7$ ,  $|C_{int}(t)| = 3$  and at least one robot position in  $C_{int}(t)$  lie on  $\mathcal{L}$ . In this case,  $\mathcal{L}$  contains either exactly one robot position or three robot positions of  $C_{int}(t)$ . Suppose that exactly one robot lies on  $\mathcal{L}$ . This robot on  $\mathcal{L}$  moves towards nearest among  $w_1$  and  $w_2$  (tie is broken arbitrarily). Other robots in  $C_{int}(t)$  move towards  $\mathcal{O}_t$ . If three robots lie on  $\mathcal{L}$ , two robots among them have free corridors to  $w_1$  and  $w_2$  and they move towards these points (Figure 4(b)). The third robot, lying on  $\mathcal{L}$ , moves towards the nearest robot position on  $\mathcal{L}$  (tie is broken arbitrarily). Let  $d_i$  be the destination point of a robot  $r_i \in C_{int}(t)$ .



**Figure 4:** An illustration of (a) special case of  $n = 8$  and (b) special case of  $n = 7$  in the algorithm *CreateMultiplicity()*

- **Case-2.3:**  $\mathcal{R}(t) \in \mathcal{QR}$

Each active robot  $r_i$  has  $c_q$  as its destination point if  $(r_i(t), c_q) \cap \mathcal{R}(t) = \emptyset$  and it moves towards this point along  $qgrad_i(t)$ . Otherwise,  $r_i$  does not move.

### 3.5 Correctness of *GatheringFault()*

---

**ALGORITHM 2:** CreateMultiplicity()

---

**Input:**  $\mathcal{R}(t)$

**Output:** A destination point.

```

if  $|C_{int}(t)| \geq \lfloor \frac{n}{2} \rfloor + 1$  then
    if  $r_i(t) \in C_{out}(t)$  then
        |    $r \leftarrow r_i(t);$ 
    else
        |    $r \leftarrow \mathcal{O}_t;$ 
    end
else
    if  $\mathcal{R}(t) \in \mathcal{QR}$  then
        if  $(r_i(t), c_q) \cap \mathcal{R}(t) = \emptyset$  then
            |    $r \leftarrow c_q;$ 
        else
            |    $r \leftarrow r_i(t);$ 
        end
    else
        if  $\mathcal{R}(t)$  is asymmetric then
            if  $r_i(t) \in \mathcal{F}(t)$  then
                |    $r \leftarrow r_i(t);$ 
            else
                |       if  $(r_i(t), w) \cap \mathcal{R}(t) = \emptyset$  then
                    |           |    $r \leftarrow \mathcal{O}_t;$ 
                else
                    |           |    $r_j(t) \leftarrow$  nearest robot position from
                    |           |    $r_i(t)$  on  $rad_i(t);$ 
                    |           |    $r \leftarrow \text{middle}\{r_i(t), r_j(t)\};$ 
                end
            end
        else
            if  $r_i(t) \in \mathcal{F}_2(t) \vee \mathcal{F}_3(t) \vee \mathcal{F}_4(t)$  then
                |    $r \leftarrow r_i(t);$ 
            else
                if  $(r_i(t), w) \cap \mathcal{R}(t) = \emptyset \wedge n \neq 7, 8$  then
                    |    $r \leftarrow \mathcal{O}_t;$ 
                else
                    |           if  $n \neq 7, 8$  then
                        |               |    $r_j(t) \leftarrow$  nearest from  $r_i(t)$  on
                        |               |    $rad_i(t);$ 
                        |               |    $r \leftarrow \text{middle}\{r_i(t), r_j(t)\};$ 
                    else
                        |               |    $r \leftarrow d_i$ 
                    end
                end
            end
        end
    end
end
end
return  $r;$ 

```

---

In this section, it is proved that the non-faulty robots shall be able to gather in finite time, by executing algorithm *GatheringFault()*, when there are at most ( $\lfloor \frac{n}{2} \rfloor - 1$ ) crashed robots and during the whole execution of the algorithm no collision occurs, where  $n \geq 7$ .

**Lemma 1.** *Algorithm MoveToDestination() provides collision free movements for the robots during the whole execution of algorithm GatheringFault().*

**PROOF.** According to algorithm *GatheringFault()*, a moving robot in  $\mathcal{R}$  has one of the followings as its destination point: (i) a specific destination point (either  $\mathcal{O}_t$  or  $c_q$  or a point on the line of symmetry) or (ii) a point inside  $SEC()$ .

Let  $r_i \in \mathcal{R}$  be a robot which moves towards its destination point, say  $w$ , according to *MoveToDestination()*. Followings are two cases in which the directions of movements of all the robots, in a particular round, are towards the same point: (i)  $(r_i(t), w) \cap \mathcal{R}(t) = \emptyset$  and (ii) a robot  $r_j \in C_{out}(t)$  wants to move inside  $SEC(t)$ . For the second case, if  $(r_j(t), w) \cap \mathcal{R}(t) \neq \emptyset$ , the robot  $r_j$  stops far enough from its next robot on the corresponding line segment  $rad_j(t)$ . Thus the robots, executing these two moves in a particular round, do not collide.

If  $(r_i(t), w) \cap \mathcal{R}(t) \neq \emptyset$ , the robot  $r_i$  shifts from the line segment  $r_i(t)w$ . This shifted destination point of  $r_i$  is in the one third sub-sector of the sector defined by  $\mu$  and closest to  $r_i(t)w$ . Thus, the movement of  $r_i$  does not obstruct the free movements of the other robots moving towards  $w$ . If  $r_i(t)w$  contains more than one robot position (excepting at  $w$ ), algorithm *MoveToDestination()* divides the closest one third sub-sector of  $\mu$  among these robots, according to their distances from  $w$ . Furthermore, the paths towards the destination points of these robots do not intersect each other (since the lines joining the robots to the destination points lie in different concentric circles centred at  $w$ ). This implies that, these robots do not collide during their shifting. At the new position  $r_i(t')$ , the robot  $r_i$  has one of the two possibilities (i)  $r_i$  has a free corridor to  $w$  or (ii)  $(r_i(t'), w) \cap \mathcal{R}(t') \neq \emptyset$ . For the second case,  $r_i$  has to shift again to find a free corridor towards  $w$ . Since there are finite number of robots, after at most finite number of shifts  $r_i$  would find a free corridor to  $w$ .

Hence, *MoveToDestination()* guarantees a collision free arrivals to the destination points for the robots.  $\square$

**Lemma 2.** *Algorithm CreateMultiplicity() creates a unique multiplicity point in finite time when the maximum number of admissible crashed robots is  $\lfloor \frac{n}{2} \rfloor - 1$  and  $n \geq 7$ .*

**PROOF.** Let us consider a robot configuration  $\mathcal{R}(t) \notin \mathcal{M}$ ,  $t \geq t_0$ .

- **Case-1:**  $\mathcal{R}(t) \in \mathcal{D}$  ( $|C_{int}(t)| \geq \lfloor \frac{n}{2} \rfloor + 1$ )

Since robots in  $C_{out}(t)$  do not move,  $SEC(t)$  does not change. There could be at most  $\lfloor \frac{n}{2} \rfloor - 1$  faulty robots in  $C_{int}(t)$ . Since  $|C_{int}(t)| \geq \lfloor \frac{n}{2} \rfloor + 1$  and  $n \geq 7$ , there are at least two non-faulty robots in  $C_{int}(t)$ . Algorithm *MoveToDestination()* guarantees collision free reachability of the robots to the destination points. According to the algorithm,  $\mathcal{O}_t$  is the destination point for each robot in  $C_{int}(t)$ . Since  $C_{int}(t)$  contains at least two non-faulty robots,  $\mathcal{O}_t$  will become a multiplicity point in finite time and by lemma 1, it is unique.

- **Case-2:**  $\mathcal{R}(t) \in \mathcal{ND}$  ( $|C_{int}(t)| < \lfloor \frac{n}{2} \rfloor + 1$ ):

- **Case-2.1 and Case-2.2:**  $\mathcal{R}(t) \notin \mathcal{QR}$  and  $\mathcal{R}(t)$  has at most one line of symmetry

First, consider the scenarios when  $n \geq 9$ . At most four non-faulty robots retain their positions in  $C_{out}(t)$  and rest of the active robots on  $C_{out}(t)$  move inside  $SEC(t)$  along the corresponding  $rad()$ . The robots in  $C_{int}(t)$  move to  $\mathcal{O}_t$  according to  $MoveToDestination()$ . Due to the movements of the robots, any one of the following scenarios will occur:

- a unique multiplicity point is created at  $\mathcal{O}_t$ .
- the condition  $|C_{int}(t')| \geq \lfloor \frac{n}{2} \rfloor + 1$  is satisfied.

We are done in of the above two cases.

- $\mathcal{R}(t')$  is Q-regular and  $|C_{int}(t')| < \lfloor \frac{n}{2} \rfloor + 1$ .

This case is discussed below.

- $\mathcal{R}(t')$  has at most one line of symmetry with  $|C_{int}(t')| < \lfloor \frac{n}{2} \rfloor + 1$ .

In this case, the robots repeat their actions.

Since, at least two robots (if there are exactly two robots, then they are diametrically opposite) retain their positions on the circumference of  $SEC(t)$ , until any one of the above scenarios (i), (ii) or (iii) is satisfied, the circle  $SEC(t)$  does not change. Since  $n \geq 9$  and there could be at most  $\lfloor \frac{n}{2} \rfloor - 1$  faulty robots, within finite time, either  $C_{int}(t)$  contains at least two non-faulty robots or  $\mathcal{O}_t$  becomes the unique multiplicity point.

Now consider the cases when  $7 \leq n \leq 8$ . The above arguments are also valid for  $7 \leq n \leq 8$  other than the special scenarios as considered in the algorithm section 3.4 (case-2.2.2). For the special scenarios, since  $C_{int}(t)$  contains at least one non-faulty robot, after a finite time, either (i) a point of multiplicity is created on  $\mathcal{L}$  by the robots on it or (ii)  $\mathcal{L}$  would contain at least one robot position in  $C_{out}(t')$ ,  $t' > t$ . The scenario (ii) is same as the case-2.2.1 in section 3.4. Thus, a unique multiplicity point would be created in finite time. Note that in the special cases, the multiplicity point could be created at the point  $w_1$  or  $w_2$ .

- **Case-2.3:**  $\mathcal{R}(t) \in \mathcal{QR}$

In this case, the centre of Q-regularity  $c_q$  is the initial destination point of the moving robot. Since  $|C_{int}(t)| < \lfloor \frac{n}{2} \rfloor + 1$ , the value of  $e_q(t)$  satisfies;  $\lfloor \frac{n}{2} \rfloor \leq e_q \leq n$ . If  $e_q = n$ , each robot has free corridor towards  $c_q$  and it moves straight towards this point. Since,  $c_q$  is the Weber point of  $\mathcal{R}(t)$  and robots move along straight lines towards it, this point remains invariant under the movements of the robots. Now, suppose that  $\lfloor \frac{n}{2} \rfloor \leq e_q(t) < n$ . Since the maximum number of faulty robots is  $\lfloor \frac{n}{2} \rfloor - 1$ , there are three possibilities: (i)  $C_{int}(t)$

contains at least two non-faulty robots which has free corridors to  $c_q$  or (ii) there is at least one  $grad_i(t)$  containing at least two non-faulty robots such that they are closest to  $\mathcal{O}_t$  than the other robot positions on  $grad_i(t)$  or (iii)  $C_{out}(t)$  contains at least one non-faulty robot which has a free corridor to  $c_q$  (e.g., all the non-faulty robots on the  $grad_i$ (s) containing multiple robot positions, are blocked by the faulty robots). In the scenario (iii), the circle  $SEC(t)$  does not change due to the movements of the robots in  $C_{out}(t)$  (a subset of robots in  $C_{out}(t)$  which do not move, are rotationally symmetric and they keep the circle intact). Thus, in finite finite, either  $c_q$  will become the unique multiplicity point or the condition  $|C_{int}(t)| \geq \lfloor \frac{n}{2} \rfloor + 1$  will be satisfied.  $\square$

**Lemma 3.** *Algorithm GatheringFault() solves the gathering problem in  $(n, \lfloor \frac{n}{2} \rfloor - 1)$  crash fault model for  $n \geq 7$  semi-synchronous robots (initially placed at different positions) with weak multiplicity detection in finite time.*

PROOF. Follows from lemma 1 and 2.  $\square$

**Theorem 1.** *The gathering problem is solvable in  $(n, \lfloor \frac{n}{2} \rfloor - 2)$  crash fault model for  $n \geq 3$  semi-synchronous robots (initially placed at different positions) with weak multiplicity detection in finite time. Further, if  $n \geq 7$ , gathering is possible in  $(n, \lfloor \frac{n}{2} \rfloor - 1)$  crash fault model.*

PROOF. If  $n \leq 6$ , the maximum number of admissible faults i.e.,  $(\lfloor \frac{n}{2} \rfloor - 2)$  is 1. The results of [2] imply that the problem is solvable for  $n \leq 6$  robots. Again, for  $n \geq 7$ , the maximum number of admissible faults is more than one and the lemma 3 implies that gathering is possible with at most  $\lfloor \frac{n}{2} \rfloor - 1$  faulty robots. Hence the theorem.  $\square$

## 4. FAULT TOLERANT ALGORITHM WITH KNOWN $\delta$

Different assumptions on the capabilities of the robots help them to solve a variety of problems. Without these assumptions, those problems are either proven impossible or no deterministic solutions are known. Whenever prices are paid in terms of these assumptions, a natural question is asked? What are we gaining in returns? In this section, we consider one of such assumption and see its effect on the solutions of the gathering problem. We assume that each robot has the knowledge of  $\delta$ . In return, it helps the robots to solve the gathering problem in  $(n, n-6)$  crash fault model with multiplicity detection capability and semi-synchronous scheduler (*SSYNC* model).

### 4.1 Configurations

We divide  $\tilde{\mathcal{R}}$  into following classes:

- **Multiple ( $\mathcal{M}$ ) :** This class contains all the configurations which have a unique multiplicity point.
- **Q-regular ( $\mathcal{QR}$ ) :** Each configuration in this class is Q-regular.
- **Dense-In ( $\mathcal{DI}$ ) :** All the configurations for which the condition  $|C_{out}(t)| \leq 4 \wedge \neg \mathcal{QR}$  is satisfied.
- **Dense-Out ( $\mathcal{DO}$ ) :** All the configurations for which the condition  $|C_{out}(t)| > 4 \wedge \neg \mathcal{QR}$  is satisfied.

The configurations in both the classes  $\mathcal{DI}$  and  $\mathcal{DO}$  have at most one line of symmetry. It is easy to see that  $\widetilde{\mathcal{R}} = \mathcal{M} \cup \mathcal{QR} \cup \mathcal{DI} \cup \mathcal{DO}$ .

#### 4.2 Algorithm $GatheringFault_\delta()$

We assume that  $n \geq 7$  and in the initial configuration  $\mathcal{R}(t_0)$ , all the robots have distinct positions. If an active robot at time  $t$  finds that  $\mathcal{R}(t) \in \mathcal{M}$ , it sets the multiplicity point  $p_m$  as its destination point. Otherwise, robots adopt algorithm  $CreateMultiplicity_\delta()$  to create a unique multiplicity point. Until a multiplicity point is created, algorithm  $CreateMultiplicity_\delta()$  maintains one of the two invariants (i) if the configuration is Q-regular, it remains Q-regular (ii) if the configuration is not Q-regular or does not become Q-regular due to the movements of the robots, the smallest enclosing circle  $SEC(t_0)$  of the initial configuration does not change. The robots move according to algorithm  $MoveToDestination_\delta()$ . This algorithm provides a collision free movements for the robots.

---

#### ALGORITHM 3: $GatheringFault_\delta()$

---

**Input:**  $r_i \in R$   
**Output:**  $r_i$  moves towards its destination.  
**if**  $\mathcal{R}(t) \in \mathcal{M}$  **then**  
  |  $r \leftarrow p_m$ ;  
**else**  
  |  $r \leftarrow CreateMultiplicity_\delta(\mathcal{R}(t))$ ;  
**end**  
Move to  $r$  using  $MoveToDestination_\delta(\mathcal{R}(t), r)$  ;

---

#### 4.3 Algorithm $MoveToDestination_\delta()$

Algorithm  $MoveToDestination_\delta()$  should provide collision free movements for the robots. The algorithm takes the advantage of known  $\delta$ . Let  $w'$  be the destination point for a robot  $r_i$  at time  $t$ . The robot  $r_i$  moves in any one of the following ways:

- If  $r_i$  has a free corridor to  $w'$  i.e.,  $(r_i(t), w') \cap \mathcal{R}(t) = \phi$ , robot  $r_i$  moves towards  $w'$  along the line segment  $r_i(t)w$ . If  $dist(r_i(t), w') \leq \delta$ , robot  $r_i$  moves directly to  $w'$  even if  $(r_i(t), w') \cap \mathcal{R}(t) \neq \phi$ .
- Suppose,  $(r_i(t), w') \cap \mathcal{R}(t) \neq \phi$ . Let  $\mathcal{L}$  be the line joining  $r_i(t)$  and  $w'$ . Let  $r_j$  be the nearest robot from  $r_i$  on  $\mathcal{L}$ , lying in between  $r_i(t)$  and  $w'$ . If  $dist(r_i(t), r_j(t)) > \frac{\delta}{2}$ , the destination point of  $r_i$  is the point  $r_i(t') \in (r_i(t), r_j(t))$  where  $dist(r_i(t'), r_j(t)) = \frac{\delta}{2}$ . If  $dist(r_i(t), r_j(t)) \leq \frac{\delta}{2}$ , robot  $r_i$  does one of the followings:
  1. If  $(r_i(t), w') \cap \mathcal{R}(t) = 1$ , robot  $r_i$  moves a distance  $minimum\{\delta, dist(r_i(t), w')\}$  along  $\mathcal{L}$  towards  $w'$ .
  2. Let  $(r_i(t), w') \cap \mathcal{R}(t) > 1$ . If  $(r_j(t), w') \cap \mathcal{R}(t) > 1$ , let  $r_k(t)$  be the nearest robot position from  $r_j(t)$  on  $\mathcal{L}$  lying in between  $r_j(t)$  and  $w'$ . Otherwise, we consider  $r_k(t) = w'$ . The destination point of  $r_i$  is computed as follows:
    - (a) If  $dist(r_j(t), r_k(t)) > \delta$ , robot  $r_i$  moves a  $\delta$  distance towards  $w'$ .

(b) If  $\frac{\delta}{2} < dist(r_j(t), r_k(t)) < \delta$ , robot  $r_i$  moves a distance  $dist(r_i(t), r_j(t)) + \frac{1}{2}dist(r_j(t), b)$ , where  $b$  is the point in  $(r_j(t), r_k(t))$  such that  $dist(b, r_k(t)) = \frac{\delta}{2}$ .

(c) If  $dist(r_j(t), r_k(t)) \leq \frac{\delta}{2}$ , robot  $r_i$  moves a distance  $dist(r_i(t), r_j(t)) + \frac{1}{2}dist(r_j(t), r_k(t))$ .

---

#### ALGORITHM 4: $MoveToDestination_\delta()$

---

**Input:**  $\mathcal{R}(t), w'$   
**Output:** Move towards the destination.  
**if**  $r_i(t) = w'$  **then**  
  |  $r \leftarrow r_i(t)$  ;  
**else**  
  | **if**  $(r_i(t), w') \cap \mathcal{R}(t) = \phi \vee dist(r_i(t), w') \leq \delta$  **then**  
    | move directly to  $w'$  along  $\overline{r_i(t)w'}$ ;  
  | **else**  
    |  $r_j(t) \leftarrow$  nearest robot position from  $r_i(t)$  on  $\overline{r_i(t)w'}$ ;  
    |  $r_k(t) \leftarrow$  nearest robot position from  $r_k(t)$  on  $\overline{r_i(t)w'}$ , if any, otherwise  $w'$ ;  
    | **if**  $dist(r_i(t), r_j(t)) > \frac{\delta}{2}$  **then**  
      |  $r \leftarrow$  the point  $u$  in  $(r_i(t), r_j(t))$  such that  
      |  $dist(u, r_j(t)) = \frac{\delta}{2}$ ;  
    | **else**  
      | **if**  $(r_i(t), w') \cap \mathcal{R}(t) = 1$  **then**  
        |  $r \leftarrow$  the point  $u'$  on  $\overline{r_i(t)w'}$  at a distance  
        |  $minimum\{\delta, dist(r_i(t), w')\}$   $r_i(t)$ );  
      | **else**  
        | **if**  $dist(r_j(t), r_k(t)) > \delta$  **then**  
          |  $r \leftarrow$  the point  $v$  on  $\overline{r_i(t)w'}$  at a  
          | distance  $\delta$ ;  
        | **else**  
          | **if**  $\frac{\delta}{2} < dist(r_j(t), r_k(t)) < \delta$  **then**  
            |  $r \leftarrow$  the point  $u'$  on  $\overline{r_i(t)w'}$  at a  
            | distance  
            |  $dist(r_i(t), r_j(t)) + \frac{1}{2}dist(r_j(t), b)$   
            | from  $r_i(t)$ , where  $b$  is the point in  
            |  $(r_j(t), r_k(t))$  such that  
            |  $dist(b, r_k(t)) = \frac{\delta}{2}$ ;  
          | **else**  
            |  $r \leftarrow$  the point  $v'$  on  $\overline{r_i(t)w'}$  at a  
            | distance  $dist(r_i(t), r_j(t)) +$   
            |  $\frac{1}{2}dist(r_j(t), r_k(t))$  from  $r_i(t)$   
          | **end**  
        | **end**  
      | **end**  
    | **end**  
  | **end**  
**end**  
move directly to  $r$  along  $\overline{r_i(t)r}$ ;

---

#### 4.4 Algorithm $CreateMultiplicity_\delta()$

- **Case-1:**  $\mathcal{R}(t)$  is Q-regular ( $\mathcal{QR}$ )

Each active robot sets  $c_q$  as its destination point. The robots which have free corridors to  $c_q$ , move towards

this point along corresponding  $q_{rad}()$ s. Otherwise, the robots follow algorithm  $MoveToDestination_\delta()$  to reach  $c_q$ .

- **Case–2:  $\mathcal{R}(t)$  is not Q-regular:** We have following scenarios:

- **Case–2.1:  $\mathcal{R}(t) \in \mathcal{DO}$  ( $|C_{out}| \leq 4$ )**

The Robots on  $C_{out}(t)$  do not move. The destination point of the robots in  $C_{int}(t)$  is  $\mathcal{O}_t$ . They follow  $MoveToDestination_\delta()$  to reach their destination points.

- **Case–2.2:  $\mathcal{R}(t) \in \mathcal{DI}$  ( $|C_{out}| > 4$ )**

Since  $\mathcal{R}(t)$  is not Q-regular,  $\mathcal{R}(t)$  has at most one line of symmetry. In this case, the robots follow same strategies as described in the algorithm  $CreateMultiplicity()$  of section 3.4 for Case–2.1 and Case–2.2 (excluding the two special cases of Case–2.2.2 when  $n = 7$  and  $n = 8$ ). However, two differences are there (i) the active robots on  $C_{out}(t)$  which want to move inside  $SEC(t)$  have  $\mathcal{O}_t$  as their destination point and (ii) the robots follow  $MoveToDestination_\delta()$  to reach their destination instead of  $MoveToDestination()$ .

---

**ALGORITHM 5:** *CreateMultiplicity $_\delta()$* 


---

```

Input:  $\mathcal{R}(t)$ 
Output: A destination point.
if  $\mathcal{R}(t) \in \mathcal{QR}$  then
    |  $r \leftarrow c_q$  ;
else
    if  $|C_{out}(t)| \leq 4$  then
        | if  $r_i(t) \in C_{out}(t)$  then
            | |  $r \leftarrow r_i(t)$ ;
        | else
            | |  $r \leftarrow \mathcal{O}_t$  ;
        | end
    else
        if  $\mathcal{R}(t)$  has no line of symmetry then
            | if  $r_i(t) \in \mathcal{F}_1(t)$  then
                | |  $r \leftarrow r_i(t)$ ;
            | else
                | |  $r \leftarrow \mathcal{O}_t$ ;
            | end
        else
            | if  $r_i(t) \in \mathcal{F}_2(t) \vee \mathcal{F}_3(t) \vee \mathcal{F}_4(t)$  then
                | |  $r \leftarrow r_i(t)$ ;
            | else
                | |  $r \leftarrow \mathcal{O}_t$ ;
            | end
        end
    end
end
return  $r$ ;

```

---

#### 4.5 Correctness of $GatheringFault_\delta()$

In this section, it is proved that  $GatheringFault_\delta()$  solves the gathering problem with at most  $(n - 6)$  faulty robots when  $n \geq 7$ .

**Lemma 4.** *Algorithm  $MoveToDestination_\delta()$  provides collision free movements during the whole execution of algorithm  $GatheringFault_\delta()$ .*

**PROOF.** Two robots collide when they meet at an undesirable point. During the execution of  $GatheringFault_\delta()$ , in a particular round, all the active robots have exactly one desired destination point, either  $c_q$  or  $\mathcal{O}_t$ . Let  $w'$  denote the destination point for the active robots in a particular round. The robots move towards  $w'$  along lines segments joining their positions to this point. Thus, the different paths of the robots towards  $w'$  meet only at this point. We show that two robots on a same path do not land up at a single point other than  $w'$  when they try to reach this point. Let  $r_i$  and  $r_j$  be two robots on a line segment  $\mathcal{L}$  joining their positions to  $w'$  such that  $dist(r_i(t), w') > dist(r_j(t), w')$ . Until,  $r_i$  crosses  $r_j$  on  $\mathcal{L}$  there is no possibility of collision. Again, if there is at least one robot position, say  $r_s(t)$ , in between  $r_i$  and  $r_j$  on  $\mathcal{L}$  or  $dist(r_i(t), r_j(t)) > \frac{\delta}{2}$ , the destination point of  $r_i$  falls into the interval  $(r_i(t), r_j(t))$ . Thus consider the case when there is no other robot positions in between  $r_i$  and  $r_j$  on  $\mathcal{L}$  and  $dist(r_i(t), r_j(t)) \leq \frac{\delta}{2}$ .

- If  $(r_i(t), w') \cap \mathcal{R}(t) = 1$ , robot  $r_i$  moves a distance  $minimum\{\delta, dist(r_i(t), w')\}$ . Also, robot  $r_j$  moves a distance  $minimum\{\delta, dist(r_j(t), w')\}$ . Since  $r_i$  and  $r_j$  are at different positions on  $\mathcal{L}$  and they move along this line segment, the only possible meeting point for them is  $w'$ .

- Let  $(r_i(t), w') \cap \mathcal{R}(t) > 1$ . Consider the point  $r_k(t)$ , as defined in the algorithm. If  $dist(r_j(t), r_k(t)) > \delta$ , robot  $r_i$  moves exactly a  $\delta$  distance from  $r_i(t)$  whereas  $r_j$  moves at least  $\delta$  distance from  $r_j(t)$ . Since  $r_i(t) \neq r_j(t)$ , they do not collide. Let  $dist(r_j(t), r_k(t)) < \delta$ . If  $dist(r_j(t), r_k(t)) \leq \frac{\delta}{2}$ , robot  $r_j$  crosses  $r_k(t)$  whereas destination point of  $r_i$  falls into the interval  $(r_j(t), r_k(t))$  and the robots do not collide. Otherwise, the destination point of  $r_i$  falls into the interval  $(r_j(t), b)$  whereas the destination point of  $r_j$  is the point  $b \in (r_j(t), r_k(t))$  where  $dist(b, r_k(t)) = \frac{\delta}{2}$ . Thus,  $r_i$  and  $r_j$  do not collide.  $\square$

**Lemma 5.** *Algorithm  $CreateMultiplicity_\delta()$  creates a unique multiplicity point in finite time when the number of faulty robots is at most  $(n - 6)$  and  $n \geq 7$ .*

**PROOF.** Let us consider a robot configuration  $\mathcal{R}(t) \notin \mathcal{M}$ ,  $t \geq t_0$ . Algorithm  $CreateMultiplicity_\delta()$  maintains one of the two invariants (i) if the initial configuration is Q-regular, it remains Q-regular (ii) otherwise, if the configuration does not become Q-regular, the smallest enclosing circle  $SEC(t_0)$  of the initial configuration remains the same. Thus, in each round, all active robots have exactly one desired destination point either  $c_q$  or  $\mathcal{O}_t$ .

- **Case–1:  $\mathcal{R}(t)$  is Q-regular ( $\mathcal{QR}$ )**

Each active robot has  $c_q$  as its destination point and it moves along corresponding  $q_{rad}()$ s towards this point. Since  $c_q$  is the Weber point of  $\mathcal{R}(t)$ , it remains invariant under the movements of the robots towards it along straight lines. The robots follow algorithm  $MoveToDestination_\delta()$  to reach  $c_q$  along corresponding  $q_{rad}()$ s and it guarantees collision free movements

of the robots. Since there are at least 6 non-faulty robots in the system and they do not wait for each other, the point  $c_q$  becomes the unique multiplicity point in finite time.

- **Case–2:  $\mathcal{R}(t)$  is not Q-regular:**

In this case, at most four active robots retain their positions on  $C_{out}(t)$  (provided  $\mathcal{R}(t)$  does not become Q-regular). Thus,  $C_{int}(t')$  will contain at least two non-faulty robots,  $t' \geq t$ . If  $\mathcal{R}(t)$  does not become Q-regular due to the movements of the robots, the circle  $SEC(t)$  remains the same (since at least two robots in  $C_{out}(t)$  which define the  $SEC(t)$ , do not move) and thus  $\mathcal{O}_t$  becomes fixed. Hence, in this case either  $\mathcal{R}(t)$  becomes Q-regular or  $\mathcal{O}_t$  becomes the multiplicity point. Algorithm  $MoveToDestination_\delta()$  guarantees the uniqueness of the multiplicity point. Note that once the configuration becomes Q-regular, it remains the same. Thus no live-lock occurs during the whole execution of  $CreateMultiplicity_\delta()$ .  $\square$

**Lemma 6.** *Algorithm GatheringFault $_\delta()$  solves the gathering problem in  $(n, n-6)$  crash fault model for  $n \geq 7$  semi-synchronous robots (initially placed at different positions) in finite time with weak multiplicity detection and the knowledge of  $\delta$ .*

PROOF. Follows from lemma 4 and 5.  $\square$

**Theorem 2.** *The gathering problem is solvable in  $(n, n-6)$  crash fault model for  $n \geq 3$  semi-synchronous robots (initially placed at different positions) in finite time with weak multiplicity detection and the knowledge of  $\delta$ .*

PROOF. If  $n \leq 6$ , the value of the number of admissible faults is 0 and by [2] the gathering problem is solvable. For  $n \geq 7$ , the lemma 6 implies that gathering is possible with at most  $(n-6)$  faulty robots. Hence the theorem.  $\square$

## 5. CONCLUSION

This paper presents two distributed gathering algorithms for semi-synchronous robots in crash fault model under following assumptions:

- When robots are endowed only with weak multiplicity detection and the system permits at most  $\lfloor \frac{n}{2} \rfloor - 1$  crash faults for  $n \geq 7$ .
- When robots are endowed with weak multiplicity detection, have the knowledge of  $\delta$  and the system permits at most  $(n-6)$  crash faults for  $n \geq 7$ .

It would be interesting to find out that whether the number of tolerable faults can be increased or the algorithms are optimal.

## 6. REFERENCES

- [1] C. Agathangelou, C. Georgiou, and M. Mavronicolas. A distributed algorithm for gathering many fat mobile robots in the plane. In *Proc. ACM Symposium on Principles of Distributed Computing*, pages 250–259, 2013.
- [2] N. Agmon and D. Peleg. Fault-tolerant gathering algorithms for autonomous mobile robots. *SIAM Journal on Computing*, 36(1):pages 56–82, 2006.
- [3] S. Bhagat, S. G. Chaudhuri, and K. Mukhopadhyaya. Fault-tolerant gathering of asynchronous oblivious mobile robots under one-axis agreement. *J. Discrete Algorithms*, 36:pages 50–62, 2016.
- [4] K. Bolla, T. Kovacs, and G. Fazekas. Gathering of fat robots with limited visibility and without global navigation. In *Proc. Swarm and Evolutionary Computation*, pages 30–38. 2012.
- [5] Z. Bouzid, S. Das, and S. Tixeuil. Gathering of mobile robots tolerating multiple crash faults. In *Proc. IEEE 33rd International Conference on Distributed Computing Systems*, pages 337–346, 2013.
- [6] Q. Bramas and S. Tixeuil. Wait-free gathering without chirality. In *International Colloquium on Structural Information and Communication Complexity.*, pages 313–327, 2014.
- [7] S. G. Chaudhuri and K. Mukhopadhyaya. Leader election and gathering for asynchronous fat robots without common chirality. *J. Discrete Algorithms*, 33:171–192, 2015.
- [8] M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro. Distributed computing by mobile robots: Gathering. volume 41, pages 829–879. 2012.
- [9] J. Czyzowicz, L. Gasieniec, and A. Pelc. Gathering few fat mobile robots in the plane. *Theoretical Computer Science*, 410(6):pages 481–499, 2009.
- [10] X. Défago, M. Gradinariu, S. Messika, and P. Raipin-Parvédéy. Fault-tolerant and self-stabilizing mobile robots gathering. In *Proc. 20th International Symposium on Distributed Computing*, pages 46–60. 2006.
- [11] P. Flocchini, G. Prencipe, and N. Santoro. *Distributed Computing by Oblivious Mobile Robots*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2012.
- [12] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer. Gathering of asynchronous robots with limited visibility. *Theoretical Computer Science*, 337(1–3):pages 147–168, 2005.
- [13] T. Izumi, Z. Bouzid, S. Tixeuile, and K. Wada. Brief announcement: The bg-simulation for byzantine mobile robots. In *Distributed Computing*, volume 6950 of *Lecture Notes in Computer Science*, pages 330–331. Springer Berlin Heidelberg, 2011.
- [14] G. Prencipe. Instantaneous actions vs. full asynchronicity: Controlling and coordinating a set of autonomous mobile robots. In *Proc. 7<sup>th</sup> Italian Conference on Theoretical Computer Science*, pages 154–171. 2001.
- [15] G. Prencipe. Impossibility of gathering by a set of autonomous mobile robots. *Theoretical Computer Science*, 384(2 - 3):pages 222–231, 2007.
- [16] I. Suzuki and M. Yamashita. Formation and agreement problems for anonymous mobile robots. In *Proc. 31<sup>st</sup> Annual Conference on Communication, Control and Computing*, pages 93 – 102, 1993.

# Fault-Tolerant Gathering of Mobile Robots with Weak Multiplicity Detection

Debasish Pattanayak  
IIT Guwahati, India  
p.debasish@iitg.ernet.in

Kaushik Mondal  
IIT Guwahati, India  
mondal.k@iitg.ernet.in

Partha Sarathi Mandal  
IIT Guwahati, India  
psm@iitg.ernet.in

H. Ramesh  
IIT Guwahati, India  
ramesh\_h @iitg.ernet.in

## ABSTRACT

There has been a wide interest in designing distributed algorithms for tiny robots. In particular, it has been shown that the robots can complete certain tasks even in the presence of faulty robots. In this paper, we focus on gathering of all non-faulty robots at a single point in presence of faulty robots. We propose a wait-free algorithm (i.e., no robot waits for other robots and algorithm instructs each robot to move in every step, unless it is already at the gathering location), that gathers all non-faulty robots in the semi-synchronous model without any agreement about the coordinate system and with weak multiplicity detection (i.e., a robot can detect if there are more than one robots at a point, but not their exact number) in the presence of at most  $n - 1$  faulty robots for  $n \geq 3$ . We show that the required capability for gathering robots is minimal in the above model, since relaxing it further makes gathering impossible to solve.

Also, we introduce an intermediate scheduling model in between the asynchronous (i.e., no instantaneous movement or computation) and the semi-synchronous (i.e., both instantaneous movement and computation) as  $ASYNC_{IC}$ , the asynchronous model with instantaneous computation. Then we propose another algorithm in  $ASYNC_{IC}$  model for gathering all non-faulty robots with weak multiplicity detection without any agreement on the coordinate system in the presence of at most  $\lfloor n/2 \rfloor - 2$  faulty robots for  $n \geq 7$  starting from any configuration with at most one multiplicity excluding  $C^*(0)$ ,  $C^*(1/k)$ ,  $C^*(1/2)$  and  $C^*(1/2 + 1/k)$ .

## CCS Concepts

• Theory of computation → Design and analysis of algorithms; Distributed algorithms; Self-organization;

## Keywords

Fault-Tolerance, Gathering, Oblivious Mobile Robots

## 1. INTRODUCTION

Distributed coordination among robots in multi-robot systems

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ICDCN '17, January 04-07, 2017, Hyderabad, India

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007786>

has garnered interest in recent years. The primary motivation in this field is to find the minimum capability required to achieve certain objectives for a system of robots. Over the course of study, various robot models have been used. Specifically, “weak robots” [6] is the most widely considered model. Weak robots are *autonomous*: behave independently, *anonymous*: do not have identifiers, *oblivious*: do not remember their past actions and *silent*: do not exchange messages with each other. Mostly, they do not follow a common coordinate system. The robots are represented as points in a plane. They may have the capability to detect *multiplicity points*, i.e. a point having multiple robots. A robot with *weak multiplicity detection* can only figure out whether a point is occupied by exactly one robot or more than one robot. Similarly with *strong multiplicity detection* a robot can detect the exact number of robots at a multiplicity point. They have either *limited* or *unlimited* visibility range. The robots are either *transparent* or *non-transparent*. Each robot follows *look-compute-move* cycles. It observes the surrounding in the *look* phase. In the *compute* phase it computes the destination based on the observation. It moves towards the destination point in the *move* phase. In the *semi-synchronous (SSYNC)* model, the global time is divided into discrete time intervals called rounds. In each round a subset of robots are activated. Once a robot is activated, it finishes one *look-compute-move* cycle in that round. The *fully-synchronous (FSYNC)* model can be considered a special case of *SSYNC*, since it activates all the robots in each round. In the *asynchronous (ASYNC)* model, any robot can be activated at any time. A robot can be idle for unpredictable but finite amount of time. We consider the scheduler to be a fair scheduler, which activates the robots infinitely many times in infinite time.

We consider mainly *crash faults*. The faults can occur due to the unreliable components in the system, which are custom made or manually built. Also sometimes there can be manufacturing defects. Apart from these, the faults can be caused by some external factor in the field of deployment. Hence there is a need to design fault-tolerant algorithms. In this paper we design the algorithms to be crash-fault tolerant.

### 1.1 Related Works

Some of the common problems for these multi-robot systems include, *leader election*: all robots agree on a leader among themselves, *gathering*: all robots gather at a single point, *convergence*: the robots come very close to each other and *pattern formation*: the robots imitate a given pattern on the plane. In *FSYNC* model, the gathering problem has been solved without making any additional assumptions to the basic model [1]. In [8], impossibility of gathering for  $n = 2$  without assumptions on local coordinate system agreement for *SSYNC* and *ASYNC* is proved. Chaudhuri

Reference	Model	# Faults	Direction	Chirality	Multiplicity Detection	Valid Initial Configuration
[1]	SSYNC	1	NO	NO	Weak	at most one multiplicity
[3]	SSYNC	$n - 1$	NO	YES	Strong	not bivalent
[4]	SSYNC	$n - 1$	NO	NO	Strong	not bivalent
Our Result	SSYNC	$n - 1$	NO	NO	Weak	at most one multiplicity
[2]	ASYNC	$n - 1$	YES	NO	NO	any
Our Result	$\text{ASYNC}_{IC}$	$\lfloor n/2 \rfloor - 2$	NO	NO	Weak	any configuration with at most one multiplicity excluding $C^*(0)$ , $C^*(1/k)$ , $C^*(1/2)$ and $C^*(1/2 + 1/k)$

Table 1: Summary of Assumptions for Fault-Tolerant Gathering

et al. [5] have proposed a deterministic algorithm for leader election and gathering for transparent fat robots without common sense of direction or *chirality*. Common chirality is basically the common clockwise order.

In recent years, devising algorithms that achieve the goal even in the presence of a few faulty robots has piqued the interest. In SSYNC model, Agmon et al. [1] have proposed an algorithm to gather robots with at most one faulty robot. Bouzid et al. [3] have proposed a wait-free crash-fault tolerant gathering algorithm with robots having strong multiplicity detection and chirality. Bramas and Tixeuil [4] have proposed a wait-free gathering algorithm for robots with arbitrary number of faults, which removed the assumption of chirality, but still has strong multiplicity detection as opposed to weak multiplicity detection in [1]. They conjecture that weak multiplicity detection can only solve gathering for distinct initial positions. Bhagat et al. [2] have solved the problem of gathering in ASYNC setting ( $n, n - 1$ ) crash fault model in 2D under agreement on the direction and orientation of one axis. To the best of our knowledge there is no fault-tolerant algorithm in the ASYNC model without any agreement in coordinate system. The various assumptions in the results we found along with our results for fault-tolerant gathering are summarized in Table 1.

## 1.2 Our Contributions

In this paper we propose two gathering algorithms, where the robots do not share a common direction (unlike [2]) or chirality (unlike [3]). The robots only have capability of weak multiplicity detection (as opposed to strong multiplicity detection in [3, 4]). The relaxation of assumption is possible because of our strategy of moving the robots in non-intersecting circular paths, which avoids creation of multiple multiplicity points. We propose a wait-free gathering algorithm, which achieves gathering in finite time without common sense of direction or chirality with only weak multiplicity detection. This algorithm extends the result by Agmon and Peleg [1] to tolerate  $n - 1$  faults. We prove the conjecture in [4] that strong multiplicity is required for gathering in presence of more than one multiplicity points in the admissible initial configurations in SSYNC model. We initiate the study on an asynchronous scheduling model with instantaneous computation ( $\text{ASYNC}_{IC}$ ) which is an intermediate model between semi-synchronous and asynchronous. We propose a fault-tolerant algorithm in the  $\text{ASYNC}_{IC}$  which can gather even if almost half the number of total robots are faulty starting from any initial configuration with at most one multiplicity other than  $C^*(0)$ ,  $C^*(1/k)$ ,  $C^*(1/2)$  and  $C^*(1/2 + 1/k)$ .

## 2. PRELIMINARIES

Some important notations are as follows.  $R = \{r_1, r_2, \dots, r_n\}$  be the set of  $n$  point robots in a Euclidean plane. Robot  $r_i$  occupies the position  $p_i$  in the plane. A multiplicity point can have

multiple robots. The configuration  $C = \{p_1, p_2, \dots, p_n\}$ , is a multiset, which represents the robots in the plane. A configuration is a *legal configuration* if it has at most one multiplicity point. The smallest enclosing circle (SEC) of a configuration  $C$  is denoted by  $\text{SEC}(C)$ .  $C^b$  denotes the set of robots on the boundary of  $\text{SEC}$ .  $C_{fix}^b$  is the set of robots on boundary which are fixed. In the  $(n, f)$  crash-fault model, out of the  $n$  robots at most  $f$  robots are faulty.

The robots have following behaviors in the crash fault model.

**Behavior 1:** For a non-faulty robot  $r_i$  at position  $p_i$  moving towards the destination  $p^*$  in its activated cycle, if the distance between  $p_i$  and  $p^*$  is less than  $S$ , where  $S$  is a constant, then the robot  $r_i$  reaches  $p^*$  in the same cycle. Otherwise the robot stops at a point on the line  $\overline{p_i p^*}$  which is at least at a distance  $S$  away from  $p_i$ .

**Behavior 2:** A robot may become faulty at any point of time.

Gathering of robots in a crash-fault model is the gathering of all non-faulty robots at one point in finite time.

## 3. IMPOSSIBILITY

It is impossible to gather two robots in SSYNC without agreement in coordinate system [8]. The adversary can always schedule the robots such that at the end there would be exactly two multiplicity points. As the robots only have capability of detecting either it is a multiplicity or not, not the capability to find out exactly how many robots are there in a multiplicity, so both multiplicity points behave the same as two robots in SSYNC model. Hence the theorem follows.

**THEOREM 1.** *For a non-legal configuration, it is impossible to design a wait-free deterministic algorithm which gathers all the robots with weak multiplicity detection in SSYNC model.*

The proof of lemmas and theorems can be found in [7].

**OBSERVATION 1.** *If any algorithm is  $n - 1$  fault-tolerant, then it must be wait-free.*

If the algorithm is not wait-free, then the non-faulty robot waits for some other robots to move. If all of them are faulty, then it results in an indefinite wait cycle. So the algorithm has to be wait-free. From theorem 1, it is clear that any wait-free gathering algorithm for SSYNC model can gather non-faulty robots only starting from a configuration with at most one multiplicity point.

## 4. GATHERING $(N, N - 1)$ CRASH FAULT IN SSYNC MODEL

**Model:** The robots are modelled as points in a Euclidean plane. Each robot can observe the environment and determine its position along with the positions of other robots in its local coordinate system. The robots are autonomous, anonymous, oblivious,

ous, homogeneous and silent. The robots have unlimited visibility. They have the capacity of weak multiplicity detection. Each robot follows an atomic *look-compute-move* cycle. Here we have considered the crash-fault model. If a robot becomes faulty then it stops functioning. The robots follow SSYNC scheduling. The problem statement is as follows:

**PROBLEM 1.** *((n, n – 1) Crash Fault): Given n anonymous, homogeneous, oblivious, point robots with unlimited visibility in a legal initial configuration with no agreement in coordinate system, having the ability to detect multiplicity points in SSYNC model. The objective is to achieve gathering of non-faulty robots in the (n, n – 1) crash fault system for n ≥ 3.*

**Algorithm and Correctness:** In this section we present Algorithm SSYNCGATHER, which gathers the non-faulty robots. In the SSYNC model, all the robots that are activated in each cycle look at the same time. Hence the view of each robot is the same. If there is a multiplicity point in the configuration then all the robots move towards the multiplicity point. If there is no multiplicity point then they move towards the center of the SEC. The robots move in a straight line path if the path is not blocked by other robots, otherwise they move in circular paths which intersect only at the destination. As any robot which is activated can move, the SEC can change in subsequent rounds. But for each round, the SEC is the same for all robots and hence the destination point (i.e., the center of SEC) also remains invariant for that round.

---

**Algorithm 1:** MOVEDEST( $C, p, p^*, style, distance$ )

---

**Input :** Robot position, destination point, movement style and distance  
**Output:** Movement of robot

```

1 if style = straight then
2   | Move from p to p* in a straight line
3 else
4   | l = FINDTANGENT(C, p, p*)
5   | Let G be the circle passing through p and p* with line l
      as a tangent to G at p*.
6   | Move from p to p* along the arc of circle G in that sector.
7 if distance = full then
8   | Move completely to p* along the path.
9 else
10  | Move along the path until the midpoint of path.

```

---

SSYNCGATHER invokes Algorithm 1(MOVEDEST), with either circular or straight path movement, given the path of the robot is blocked or not. Algorithm 1 uses Algorithm 2 (FINDTANGENT) as a subroutine to find a line, which should be tangent to the circular path. The circular path is calculated by taking the angle bisector of the smallest angle formed by another robot with respect to the center of the SEC as the tangent to the circle passing through the robot and center of SEC. If there are multiple robots on a single line, say at  $p_1, p_2$  and  $p_3$ , in the order towards the goal, then  $p_2$  finds the position of  $p_1$  in the next round. If  $p_1$  gets activated, then it uses the line joining  $p_1$  and the destination as the tangent to its circular path. Note that, this is a recursive process for all the robots in between the first and last robot in the same line, starting from the last but one. Once the multiplicity point is formed, all robots move towards the multiplicity point, which remains invariant until gathering is finished in a similar manner.

**LEMMA 1.** *If the center of SEC is different in the two consecutive cycles, then the new radius is less than the previous one.*

---

**Algorithm 2:** FINDTANGENT( $C, p, p^*$ )

---

**Input :** Robot position, destination and configuration  
**Output:** A line

```

1 if there is a robot at p' next to p on  $\overrightarrow{p^*p}$  outside  $\overrightarrow{p^*p}$  then
2   | l = FINDTANGENT( $C, p', p^*$ )
3   | Let circle G passes through p' and p* with l as a tangent
      at p*.
4   | if arclength of the minor arc is less than S then
5     |   | Return l
6   | else
7     |   | Let p'' be the point on G such that length of  $\overrightarrow{p'p''}$  is S
8     |   | Return  $\overrightarrow{p''p^*}$ 
9 else
10  | Find the robot free sector of SEC adjacent to p with
      smallest angle, say  $\mu$ .
11  | Return the angle bisector of  $\mu$ .

```

---

**LEMMA 2.** *If a robot moves distance  $\delta$  towards the center of SEC radially, then the distance from the center of SEC in the next cycle is at most  $\frac{r - \delta}{2} + \frac{1}{2}\sqrt{(r + \delta)^2 - 2\delta^2}$ , where r is the radius of SEC in the current cycle.*

**LEMMA 3.** *For any non-faulty robot, the minimum radial distance traveled towards the center of SEC in one cycle is at least  $\lambda$ , where  $\lambda > \frac{S^2}{2r}$ .*

**THEOREM 2.** *For a legal initial configuration, SSYNCGATHER gathers the non-faulty robots in finite time for  $n \geq 3$ .*

## 5. GATHERING ( $N, \lfloor N/2 \rfloor - 2$ ) CRASH FAULT IN ASYNC<sub>IC</sub> MODEL

**Instantaneous Computation:** In ASYNC model, there can be arbitrarily large delay between the *look*, *compute* and *move* phases. Thus a robot can move based on an outdated look state. This can cause algorithms to fail. One can construct an example which shows the downside of execution of an algorithm based on an outdated look state. Now consider there is no computation delay. Then the robots would always move based on the current look information. Then situations like the previous one can be avoided. So we in this paper introduce the asynchronous model with instantaneous computation (ASYNC<sub>IC</sub>). In ASYNC<sub>IC</sub> the computation phase is instantaneous. This means that there is no delay between look and move. Any robot after completion of look stage, immediately starts moving towards the destination. For example, if two robots look at time  $t$  and  $t'$ , where  $t' = t + \epsilon$  for some small  $\epsilon > 0$ , the robot looked at time  $t$  has already started its movement (unless the destination computed is itself) by the time the second robot looks at  $t'$ . This model is denoted as ASYNC<sub>IC</sub>. The inactivity period is unpredictable but finite. Since the conflict cannot happen due to compute delays, the ASYNC<sub>IC</sub> model can be considered more powerful than ASYNC but less powerful than SSYNC.

**Model:** The robots considered in this section have exactly the same capabilities as in section 4, i.e., they are autonomous, anonymous, homogeneous, oblivious, silent and have weak multiplicity detection. The scheduling model followed here is the ASYNC<sub>IC</sub>. In this section, gathering problem is solved for ( $n, \lfloor n/2 \rfloor - 2$ ) crash fault in the ASYNC<sub>IC</sub> model.

**PROBLEM 2.**  $((n, \lfloor n/2 \rfloor - 2) \text{ Crash Fault})$ : Given  $n$  anonymous, homogeneous, oblivious, point robots with unlimited visibility in legal initial configuration with no agreement in coordinate system, having the ability to detect multiplicity points in ASYNC<sub>IC</sub> model. The objective is to achieve gathering for all non-faulty robots in  $(n, \lfloor n/2 \rfloor - 2)$  crash fault system for  $n \geq 7$ .

**Algorithm and Correctness:** Now we present Algorithm ASYNC-GATHER. ASYNCGATHER is executed in such a fashion that the SEC of the initial configuration remains as the SEC of all the configurations to follow until a multiplicity point is formed. Initially, let  $k$  robots are on the boundary and remaining  $n - k$  robots are inside the SEC. If  $k \leq n/2$ , fix all the robots which are on the boundary of the SEC. So there are at least two non faulty robots which move towards the center of the SEC and create a multiplicity point. If  $k > n/2$ , we divide the SEC into  $k$  cells by taking the bisector of the chords joining two consecutive robots on the boundary and then use pigeonhole principle to conclude that there exists at least an empty cell since  $n - k < k$ . Among  $k$  cells, let  $k_1$  are empty cells. If  $k_1 \leq k/2$ , we fix robots corresponding to the empty cells. Else if  $k_1 > k/2$ , we fix robots corresponding to non empty cells. In any case we are fixing at most  $k/2$  robots. So there are at least two non faulty robots which move towards the center of the SEC and create a multiplicity point. All configurations, where fixing the SEC is possible after making cells due to the presence of both empty and non-empty cells, are denoted by  $C(\text{Cell})$ .  $C(\text{Cell})$  may contain symmetric as well as asymmetric configurations.

There are configurations for which fixing the SEC is not possible using the above strategy. These configurations have the same number of robots in each cell since robots may lie on cell boundary and consequently each cell may contain the same number of robots. The robots present on the cell boundary are considered to be equally shared between the cells. If a robot lies on the boundary of two cells, then it is counted as 0.5 per cell. Say  $v_i$  be the number of robots present in each cell. The possible configurations with all the cells having same  $v_i$  where  $k > n/2$  and no multiplicity point present are following.

For  $i \in \{1, 2, \dots, n\}$   $C(0) : v_i = 0$ ,  $C(1/k) : v_i = 1/k$  i.e.,  $k$  robots are on the boundary and one robot is at the center,  $C(1/2) : v_i = 1/2$ ,  $C(1/2 + 1/k)$ : Combination of  $C(1/k)$  and  $C(1/2)$  configuration. These configurations can be symmetric as well as asymmetric. If any of the  $C(0)$ ,  $C(1/k)$ ,  $C(1/2)$  or  $C(1/2 + 1/k)$  configuration is asymmetric then we can elect a leader. Consequently we can fix the SEC according to FIXSEC. We use the leader election algorithm presented in [5], which deterministically elects a leader for any asymmetric configuration. We show that these configurations can only be the initial configuration. If any robot moves according to our algorithm from any of these configuration then it reaches  $C(\text{Cell})$ . Once the configuration is in  $C(\text{Cell})$ , it never reaches any of the four configurations stated above. Henceforth symmetric configurations are represented by  $C^*(\cdot)$ . If the initial configuration is symmetric as well as any of the  $C(0)$ ,  $C(1/k)$ ,  $C(1/2)$  or  $C(1/2 + 1/k)$ , our algorithm cannot gather since leader election is not possible. So these four class of symmetric configurations, i.e.,  $C^*(0)$ ,  $C^*(1/k)$ ,  $C^*(1/2)$  or  $C^*(1/2 + 1/k)$  are not included in the admissible initial configuration.

For a given configuration  $C = \{p_1, p_2, \dots, p_n\}$ , the SEC of  $C$  can be represented by a set  $C^b$  of points which lie on the SEC. The SEC of a set of points can be represented with a subset of those points. If all the other robots, which are not part of that subset move inside the SEC, the SEC remains the same. The idea behind FIXSEC (FixSEC( $F, C$ )) is to choose a subset of robots such that the SEC remains invariant as long as the robots in that sub-

set do not move.

**OBSERVATION 2.** In the set  $C^b$  either there are two points which form a diameter or there are at least three points which do not lie on a semicircle.

According to Observation 2, FIXSEC can always choose two or three robots for fixing the SEC. The robots to be fixed are chosen from either the empty cells or the non-empty cells whichever is minimum. If all the robots are on the same semicircle, then we have to choose the farthest two and take the robots closest to the chord bisector on the other semicircle. In this manner, we can choose at most four robot to fix. If the robots chosen are not present in a semicircle, then we fix all of them, resulting in at most  $k/2$  fixed robots.

**LEMMA 4.** FIXSEC determines a subset  $C_{fix}^b$  of  $C^b$  such that SEC of  $C_{fix}^b$  is the SEC of  $C$ .

**THEOREM 3.** ASYNCGATHER gathers all non-faulty robots in finite time for  $n \geq 7$  from any configuration with at most one multiplicity excluding  $C^*(0)$ ,  $C^*(1/k)$ ,  $C^*(1/2)$  and  $C^*(1/2 + 1/k)$ .

## 6. CONCLUSION

In this paper, we have solved the problem of gathering in finite time for  $(n, n - 1)$  crash fault in SSYNC model without any assumption on the coordinate system. This answers the open question in [4] for fault-tolerant gathering in SSYNC model. In ASYNC<sub>IC</sub> model under similar setting, we have solved the gathering problem with at most  $\lfloor n/2 \rfloor - 2$  faulty robots.

We think the ASYNC<sub>IC</sub> model is a step towards the positive direction. There are many problems which are still unsolved or proved to be unsolvable in ASYNC model can be addressed in the ASYNC<sub>IC</sub> model. Many other problems like pattern formation, flocking, etc., can also be addressed in ASYNC<sub>IC</sub> model.

## 7. REFERENCES

- [1] Noa Agmon and David Peleg. Fault-tolerant gathering algorithms for autonomous mobile robots. *SIAM J. on Computing*, 36(1):56–82, 2006.
- [2] S. Bhagat, S. Gan Chaudhuri, and K. Mukhopadhyaya. Fault-tolerant gathering of asynchronous oblivious mobile robots under one-axis agreement. *Journal of Discrete Algorithms*, 36:50 – 62, 2016.
- [3] Zohir Bouzid, Shantanu Das, and Sébastien Tixeuil. Gathering of mobile robots tolerating multiple crash faults. In *ICDCS*, pages 337–346, 2013.
- [4] Quentin Bramas and Sébastien Tixeuil. Wait-free gathering without chirality. In *Proc. of SIROCCO*, pages 313–327, 2015.
- [5] Sruti Gan Chaudhuri and Krishnendu Mukhopadhyaya. Leader election and gathering for asynchronous fat robots without common chirality. *J. Discrete Algorithms*, 33:171–192, 2015.
- [6] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Peter Widmayer. Hard tasks for weak robots: The role of common knowledge in pattern formation by autonomous mobile robots. In *Proc. ISAAC*, pages 93–102, 1999.
- [7] Debasish Pattanayak, Kaushik Mondal, H. Ramesh, and Partha Sarathi Mandal. Fault-tolerant gathering of mobile robots with weak multiplicity detection. *CoRR*, abs/1608.02432, 2016.
- [8] Ichiro Suzuki and Masafumi Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM J. on Computing*, 28(4):1347–1363, 1999.

# Time-communication trade-offs for minimum spanning tree construction

Ali Mashreghi \*

Department of Computer Science  
University of Victoria, BC, Canada  
ali.mashreghi87@gmail.com

Valerie King \*

Department of Computer Science  
University of Victoria, BC, Canada  
val@uvic.ca

## ABSTRACT

This paper concerns the problem of constructing a minimum spanning tree (MST) in a synchronous distributed network with  $n$  nodes, where each node knows only the identities of itself and its neighbors. We assume the CONGEST model where messages are of size  $O(\log n)$  bits. Spanning tree construction was long believed to require an amount of communication linear in the number of edges. In 2015, King, Kutten and Thorup presented a Monte Carlo algorithm which broke this communication bound. In particular it showed that an MST could be constructed with time and message complexity  $O(n \log^2 n / \log \log n)$ , independent of the number of edges.

Here we give trade-offs between time and communication. Our Monte Carlo algorithm runs in  $O(n/\epsilon)$  time and  $O(\frac{n^{1+\epsilon}}{\epsilon} \log \log n)$  messages for any  $1 > \epsilon \geq \log \log n / \log n$ . For the spanning tree problem, we show a time bound of  $O(n)$  and a communication bound of  $O(n \log n \log \log n)$  messages. We also provide the first algorithm that constructs an MST in time proportional to the diameter of the MST up to a logarithmic factor with  $o(m)$  communication.

## Keywords

CONGEST; Minimum spanning tree; Randomization; Time complexity; Message complexity

## 1. INTRODUCTION

The problem of broadcasting over a distributed network algorithms efficiently in terms of time and communication is a fundamental problem which has been considered by many researchers over the last 35 years. If we consider the network as an undirected graph with  $n$  nodes and  $m$  edges, then the spanning tree (or broadcast tree) allows one node to broadcast to  $n - 1$  other nodes in a network using only  $n - 1$  messages. If each link is weighted, e.g., by the energy

\*Funded with an NSERC discovery grant Algorithm Design for Large Graphs and Communications Networks.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

ICDCN '17 January 04-07, 2017, Hyderabad, India

© 2017 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-4839-3/17/01.

DOI: <http://dx.doi.org/10.1145/3007748.3007775>

needed to send a message over the link, then the minimum spanning tree gives a way to perform the broadcast using the least amount of energy. We show how to efficiently construct a minimum spanning tree (or forest, if the graph is not connected) in a distributed network. Our algorithms are Monte Carlo, and succeed with high probability in the synchronous CONGEST model, where each node may send a message of  $O(\log n)$  bits to each of its neighbors in a single round.

A simple algorithm for the spanning tree problem is to construct a breadth-first search tree, by having the root node send to its neighbors and these nodes send to their neighbors, etc., adding an edge to each newly discovered node. This takes optimal time, proportional to the graph diameter, and uses  $O(m)$  communication. The classic algorithm by Gallager, Humblet and Spira (GHS) for finding an MST in a distributed (asynchronous) [GHS83] takes  $O(n \log n)$  rounds and uses  $O(m + n \log n)$  messages. A lower bound of  $m$  bits of communication was long conjectured, and has been shown in various models. This bound does not apply to the "standard" model, also called the  $KT_1$  model in Peleg [Pel00], where each node knows its unique identity and those of its neighbors and the weights of the edges incident to it, *if* identities can be manipulated arbitrarily. In 2015, King, Kutten and Thorup (KKT) [KKT15] gave Monte Carlo algorithms which rely on performing simple arithmetic operations on identities to go below the bound with both message and time complexity  $\tilde{O}(n)$ . In particular, their algorithm for minimum spanning tree has time and message complexity  $O(n \log^2 / \log \log n)$ , independent of the number of edges.

The running time of GHS was improved by Chin and Ting, [CT85], Gafni [Gaf85], and then Awerbuch [Awe87], who found an  $O(n)$  time algorithm with  $O(m)$  messages. In GHS and these algorithms, nodes are merged to build increasingly large subtrees of the MST, known as *fragments*. A node determines if an incident edge is outgoing from its current fragment by querying the other endpoint of the edge, where the incident edges are considered sequentially in increasing order of weight. Therefore, finding an outgoing edge from a fragment may take a time proportional to the size of the fragment. This then is a bottleneck for the running time of these techniques, which can be as large as  $n$  and much larger than the diameter of the MST.

An alternative approach is for a node to query these edges in parallel but this results in increased communication. A series of works reduced time to  $o(n)$  but with  $\omega(m)$  messages. These make use of a breadth-first search tree to communicate quickly. Let  $D$  is the network's diameter:

Garay et al. [GKP98] introduced an  $O(D + n^{0.614})$  time algorithm. Kutten and Peleg [KP95] improved this to time  $O(D + \sqrt{n} \cdot \log^* n)$ . In the LOCAL model where messages are of unbounded size, Elkin [Elk04], pointed out another method where each edge is checked to determine if it is the heaviest on any cycle. This takes time proportional to the  $MST - radius$ , which is the maximum over all non-MST tree edges  $e$ , of the minimum length of a cycle in which  $e$  is the heaviest edge; it may be much smaller than  $D$ . Elkin [Elk04] presented an  $\tilde{O}(MST - radius + \sqrt{n})$  time algorithm for the CONGEST model. Recently, Pandurangan et al. [PRS16] gave an algorithm that simultaneously matches the time lower of  $\tilde{\Omega}(D + \sqrt{n})$  [Elk06] and the message lower bound of  $\Omega(m)$  [KPP<sup>+</sup>15] in the  $KT_0$  model.

Our paper explores the question of whether MST construction can be sped up to linear or sublinear time in  $n$  and still use only  $o(m)$  messages. We show how to construct an MST in  $\tilde{O}(\text{diam}(\text{MST}))$  time with  $\tilde{O}(n)$  messages, where  $\text{diam}(\text{MST})$  denotes the maximum hop diameter (unweighted diameter) of any MST in the network. When  $\text{diam}(\text{MST})$  is  $o(\sqrt{n})$ , this is less than the lower bound for time of  $\Omega(D + \sqrt{n}/\log n)$  [SHK<sup>+</sup>12] in the CONGEST model, which holds for Monte Carlo algorithms. However, our result does not contradict this lower bound, as the examples used to establish that bound have MSTs with diameter  $\sqrt{n}$ . Note that GHS can be modified to run in  $\tilde{O}(\text{diam}(\text{MST}))$  time with  $O(m)$  messages (see [PRS16]). Our key idea is to use the KKT method to find outgoing edges which avoids the time/communication bottleneck described above.

## 1.1 Other related work

A similar model to CONGEST model is the *Congested Clique* model. In this model, nodes in the network can directly communicate to each other through a complete underlying graph. Graph connectivity and minimum spanning tree problems have been studied in this model as well. Lotker et al. [LPSPP05] which provided an algorithm to compute the MST in  $O(\log \log n)$  rounds. This was improved to an  $O(\log \log \log n)$ -round algorithm due to Hegeman et al. [HPP<sup>+</sup>15]. Their algorithm is a randomized Monte Carlo which uses *Linear Sketches* technique to encode the neighborhood of each node, in a manner similar to MST-KKT. Ghaffari and Parter [GP16] have recently improved the time to  $O(\log^* n)$  rounds. The reason for this low number of rounds is that the diameter of the communication network is one and information can be quickly transferred. Therefore, following these ideas directly in the simple CONGEST model will introduce a factor of  $O(n)$  to the time complexity. The two mentioned algorithm still use  $\Theta(n^2)$  messages. However, in [HPP<sup>+</sup>15] an algorithm is presented that uses  $O(n \cdot \text{polylog}(n))$  messages and  $O(\text{polylog}(n))$  rounds. The algorithms for the *Congested Clique* model do not seem to provide a way to find a linear or near linear time algorithm for MST construction with  $o(m)$  messages in the general CONGEST model.

## 1.2 Our Results

We prove the following theorems in the synchronous CONGEST model where each node knows its neighbors and the weights of edges incident to it, and integer weights no greater than  $n^k$  for  $k$  a constant.

**THEOREM 1.** *An MST can be constructed with high probability in a network with  $n$  nodes in*

1.  $O(\text{diam}(\text{MST}) \frac{\log^2 n}{\log \log n})$  time and  $O(n \frac{\log^2 n}{\log \log n} \log \text{diam}(\text{MST}))$  messages;
2.  $O(n/\epsilon)$  time and  $O((1/\epsilon)n^{1+\epsilon} \log \log n)$  messages where  $\log \log n / \lg n \leq \epsilon < 1$ .

According to the second part of this theorem, choosing a constant value for  $\epsilon$ , e.g. 0.1, we can have linear time and  $O(n)$  and  $O(n^{1.1} \log \log n)$  messages which could be significantly less than the number of edges.

**THEOREM 2.** *A spanning tree can be constructed with high probability in a network with  $n$  nodes in  $O(n)$  time and  $O(n \log n \log \log n)$  messages.*

## 1.3 Overview of the algorithms

First we define some important terms. At any time step of the algorithms a *fragment* is a subtree of the final MST. Each fragment has a leader which could be any node on that fragment, e.g. the node with smallest ID. *Height* of a fragment is the maximum number of edges between a node and the leader on that fragment.

Also we define  $\text{diam}(\text{MST})$  to be the hop diameter of the MST which is the maximum number of edges on any path in the MST. Hop diameter of a minimum spanning forest is the maximum hop diameter over all of its spanning trees. Let KKT denote the MST algorithm in [KKT15], the MST is built up in iterations, from a set of fragments which are disjoint subtrees of the MST. Initially, each node is a fragment. In each iteration, fragments find outgoing edges with minimum weight and merge through these outgoing edges until the MST is found. We observe that the time to find a minimum weight outgoing edge in one fragment is proportional to the *height* of the fragment (times  $O(\frac{\log n}{\log \log n})$ ). Height of a fragment is defined to be . The KKT runs for  $O(\log n)$  iterations; each iteration is assumed to require the worst case time, i.e., that the height of some fragment in the iteration is  $n$ .

To achieve  $\tilde{O}(\text{diam}(\text{MST}))$  time for constructing an MST, it suffices to alter KKT so that it is run in phases, with increasing larger guesses as to the diameter of the MST. Initially, the diameter of the MST is assumed to be a constant. Each fragment verifies with high probability that it has no outgoing edges. If it does, the diameter is doubled and the algorithm is repeated.

To achieve  $O(n/\epsilon)$  time for finding the MST is a bit more complicated. We modify KKT so that an iteration on a fragment of height  $T$  takes time proportional to its height. We apply a threshold  $T$  so that only fragments that have height less than  $T$  are permitted to look for outgoing edges. Once enough iterations are executed so that all fragments with height less than  $T$  are merged, we are left with a small number of fragments. This allows us to apply fewer iterations to merge them. We repeat until the final MST remains. The algorithm runs in less than  $\log^* n$  phases. For the first phase  $i = 1$ , we apply a threshold of  $T = n/(\log n)^2$  and run for  $\log n$  iterations. For the following phases  $i = 2, 3, \dots, \log^* n$ , we apply a threshold  $T(i) = \frac{n}{(\log^{(i)})^2}$  and run  $O(\log^{(i)})$  iterations, where  $\log^{(i)} n$  denotes  $i$  iterated logarithms. Each iteration requires  $n/\epsilon$  time and the total time is therefore  $O((n/\epsilon) \sum_i 1/\log^{(i)})$ .

The thresholding idea is implemented in a manner similar to [Awe87]. There are two other key points here:

1. The time complexity of the KKT is  $O(n \log n \frac{\log n}{\log \log n})$ . The application of thresholds removes one  $\log n$  factor. The factor  $O(\frac{\log n}{\log \log n})$  is due to a  $\log n$ -ary search for finding the lightest outgoing edge. In order to reach the overall time complexity, we modify FINDMIN-C from KKT to run an  $n^\epsilon - ary$  search using a *pipelined broadcast-and-echo* which increases communication by an  $O(n^\epsilon / \log n)$  factor and replaces the  $O(\frac{\log n}{\log \log n})$  time factor by  $O(1)$  for constant  $\epsilon$ .

2. In each iteration of KKT, there is a constant probability of reducing the number of fragments by a constant factor. An expected  $O(\log n)$  iterations are needed to construct the MST. With an additional  $O(\log n)$  iterations, KKT succeeds with high probability. In order to get high probability of merging all trees with heights below  $T$  for each phase, we use  $O(\log n)$  parallel repetitions of each iteration instead of just one. This does not affect the search time; however, it increases communication by a factor of  $\log n$ .

We construct a spanning tree in linear time but with less communication than we use for MST construction. As in the MST construction, we use gradually increasing thresholds on the amount of time for each iteration, but in the spanning tree, outgoing edges may form a cycle of fragments. The problem of breaking the cycle in time proportion to the threshold requires a somewhat technical solution.

## 1.4 Organization of the paper

Section 2 describes the preliminaries and reviews the algorithms of [KKT15]. Section 3 shows the modified version of KKT which works in  $\tilde{O}(\text{diam}(MST))$  time rounds. Section 4 provides the linear time algorithm minimum spanning tree construction. In Section 5, the spanning tree algorithm is described. In Section 6, the conclusion and open problems are presented.

## 2. PRELIMINARIES

### 2.1 The model

The network is an undirected graph  $G = (V, E)$  in which  $|V| = n$ ,  $|E| = m$ , and messages of  $O(\log n)$  bits are allowed. We assume the CONGEST model: Communication proceeds in *synchronous* rounds (also known as time steps); a message that is sent through a link at the beginning of the round will be received by the end of the round; and in one round a node may send a message to each of its neighbors. Initially each node knows only its own ID plus the the ID's of its neighbours and the weights of the edges to its neighbors ( $KT_1$  model in [Pel00]).

All nodes know the same upper bound on the network size, i.e.,  $n$ , and all nodes start at time 0. Edge weights are positive integers less than  $n^k$  where  $k$  is any constant. Note that edge weights can be made unique simply by concatenating the ID's of the nodes to end of edge weights; hence for the remainder of the paper, we assume they are unique. As a result, our network has a unique MST.

### 2.2 Review of KKT and ST-KKT

Throughout the paper KKT and ST-KKT refer to the algorithms in [KKT15] for minimum spanning tree and spanning tree, respectively. These algorithms rely on a basic observation: Every edge with both endpoints in a fragment contributes 2 to the sum of degrees of nodes in the fragment, while an outgoing edge contributes 1. Hence if the parity of this sum is odd, there must be an outgoing edge.

Here we describe some subroutines from KKT [KKT15] that we will use directly or modify.

**Broadcast-and-echo( $F$ ):** A basic communication pattern in which a leader of a fragment  $F$  broadcasts a message to all nodes in the fragment (*broadcast*) and in response all the nodes (starting from leaves) send up a message towards the leader (*echo*).

**Odd hash function** A function  $h : [1, m] \rightarrow \{0, 1\}$  such that for any non-empty set  $S \subseteq [1, m]$ , with probability  $1/8$ , hashes an odd number of members from  $S$  to 1. In particular, let us say that for some fragment,  $S$  is the multiset of all edges connected to the nodes of that fragment. If this fragment has at least one outgoing edge then with probability  $1/8$ ,  $\sum_{e \in S} h(e) \bmod 2 = 1$ . This is because any edge whose both endpoints are in the fragment will appear twice and will not contribute to the sum. Therefore, by computing this sum in a fragment with constant probability we can say whether it has an outgoing edge.

**TestOut( $x, I$ ):** With constant probability, returns true if there is an outgoing edge leaving  $F_x$  (the fragment of leader  $x$ ) whose weight is in interval  $I$ , and false otherwise. TESTOUT uses an *odd hash function* to test if there is an outgoing edge in the given interval.

**HP-TestOut( $x, I$ ):** Does the same thing as TestOut except with high probability.

**FindMin-C( $x$ ):** Finds the lightest outgoing edge from a fragment  $F_x$  with constant probability, using worst case  $O(|F_x| \log n / \log \log n)$  messages and time. If there is no edge leaving the tree FINDMIN-C always returns  $\emptyset$ . It works by partitioning the whole search range ( $n^k$ ) each time into  $\log n$  equal sized intervals and obtaining the result of TESTOUT for each interval. Then the first interval that has an outgoing edge is picked and partitioned again. Repeating this, after  $\log_{\log n} n^k = O(\frac{\log n}{\log \log n})$  iterations the lightest outgoing edge is found.

**FindAny-C( $x$ ):** Finds *any* outgoing edge from a fragment  $F_x$  with constant probability, using worst case  $O(|F_x|)$  messages and time (one broadcast-and-return). If there is no edge leaving the tree, it always returns  $\emptyset$ . We note that the broadcast requires a  $O(\log n)$  bit transmission of a hash function and the return requires the return of an  $O(\log n)$  bit word.

KKT works in  $O(\log n)$  phases. In each phase, every fragment leader uses FINDMIN-C to find the lightest outgoing edge with constant probability. Afterwards, the fragments merge using the lightest outgoing edges they have found. This is shown to divide the number of fragments by a constant each time; hence, having  $O(\log n)$  phases overall. FINDMIN-C works by broadcasting an odd hash function and an interval which is to be divided to  $O(\log n)$  equal sized subintervals. For each subinterval in parallel, TESTOUT is applied to find the first subinterval that has an outgoing edge. Therefore, the dividing process is repeated for this subinterval until the minimum outgoing edge is located. HP-TESTOUT is used to verify the results returned by TESTOUT.

### 3. MST WITH SMALL DIAMETER

In this section we present a modification of KKT which takes  $O(\text{diam}(\text{MST}) \frac{\log^2 n}{\log \log n})$  time and uses sublinear communication in the number of edges when  $m$  is sufficiently large.

Each iteration of KKT allows the fragments  $O(n)$  time rounds to look for outgoing edges. However, we know that over the course of algorithm each fragment is actually a part of the final MST. This implies that if the diameter of the MST is low then the diameter of any fragment is low as well. In  $O(\text{diam}(\text{MST}))$  time fragments can find lightest outgoing edges. However, we do not know MST's diameter beforehand; hence, we have to guess. In particular, we start from a constant estimate of the diameter of the MST, and simulate a version of KKT assuming our estimation is an upper bound on diameter. We find the MST as soon as our estimation  $\hat{D}$  for  $\text{diam}(\text{MST})$  is greater than  $\text{diam}(\text{MST})$ . When we estimate  $\hat{D}$ , we prevent fragments from spending time more than  $O(\hat{D})$ .

We use the thresholding technique to distinguish between fragments with height larger than the desired threshold  $T$  on the height of fragments and those with height less than  $T$ . Let *active fragments* be those allowed to look for an outgoing edge, i.e., those with height less than  $T$ . For the purpose of distinguishing, before starting an iteration of KKT we do as follows.

The algorithm runs in iterations which are timed according to the global clock. At the start of each iteration all leaders deactivate themselves. Every leaf makes a *timer message* and sends it upward. A *timer* is a message with the initial value of threshold  $T$  and each time it passes a link its value is decreased by one, and once the value hits zero it will stop being transmitted. Timer message is very similar to the *exploration token* used by Awerbuch in [Awe87]; however, here, a timer does not try to measure size of a fragment and only cares about the height of a fragment which is the deciding factor for time complexity.

When all the leaves (in all fragments) sent up their timers, every internal node waits for the timers from all its children and when it has received all of them, it sends up the timer with minimum value. Consequently, the leader of each fragment will receive the timers of all of its children *if and only if* the height of the fragment is no greater than the threshold. If the height of the fragment is no more than  $T$  the leader will be activated (along with its fragment) and it can start to look for outgoing edges in the upcoming KKT iteration. Algorithm 1 shows a pseudo-code of this activation process.

Now using the ACTIVATE procedure we can present the algorithm DIAM-KKT which only takes  $O(\text{diam}(\text{MST}) \frac{\log^2 n}{\log \log n})$  time. The algorithm starts by setting the estimation of  $\text{diam}(\text{MST})$ , i.e.  $\hat{D}$ , to 1. Afterwards, before running an iteration of KKT it first activates only fragments whose height is less than  $\hat{D}$ . Intuitively, we are assuming that  $\hat{D}$  is the real value of  $\text{diam}(\text{MST})$ ; hence, there is no point in allowing fragments with height bigger than  $\hat{D}$  to look for outgoing edges. Knowing this, if we were proved wrong and could not find the MST we will make a higher estimation by doubling  $\hat{D}$ . After performing enough iterations, i.e.  $O(\log n)$ , we need to see if we could find the minimum spanning tree/forest or not. To do this, we actually do not need to know the exact size of network; we can

test w.h.p the existence of outgoing edges using a verification method. As soon as our estimate is good enough, i.e.  $\text{diam}(\text{MST}) \leq \hat{D} < 2 \cdot \text{diam}(\text{MST})$ , w.h.p the MST is found. If the minimum spanning tree or forest is found, in step 9 of DIAM-KKT, no fragment can have outgoing edges; therefore, all nodes of the network receive the message STOP and the algorithm successfully terminates. Next theorem will bound time and message complexity of DIAM-KKT.

**THEOREM 3.** DIAM-KKT uses  $O(\text{diam}(\text{MST}) \frac{\log^2 n}{\log \log n})$  time and  $O(n \frac{\log^2 n}{\log \log n} \cdot \log \text{diam}(\text{MST}))$  messages w.h.p.

**PROOF.** W.h.p the algorithm finishes as soon as  $\hat{D} \geq \text{diam}(\text{MST})$ . Thus, w.h.p. we have a total of  $\log \text{diam}(\text{MST})$  estimations since each time we are doubling  $\hat{D}$ , and factor  $\log \text{diam}(\text{MST})$  appears in message complexity. However, in the time complexity, apart from the log factors, each time we are doubling the time allowed for communication. Therefore, sum of this geometric sequence is dependent on the last term which is, w.h.p. at most twice the real value of the MST's diameter, i.e.  $2 \cdot \text{diam}(\text{MST})$ . Besides, we have  $O(\log n)$  iterations, and a multiplicative factor of  $O(\frac{\log n}{\log \log n})$  due to the  $\log n$ -ary search of FINDMIN-C. Hence, the overall time complexity is  $O(\text{diam}(\text{MST}) \frac{\log^2 n}{\log \log n})$ .  $\square$

### 4. LINEAR TIME ALGORITHM FOR CONSTRUCTING THE MST

DIAMKKT is a very fast algorithm if the diameter of the MST is small enough. However, if  $\text{diam}(\text{MST}) > \frac{n \log \log n}{\log^2 n}$  then it will take  $\omega(n)$  time rounds. Here, we want to present an algorithm for MST construction which *always* works in linear time and takes sublinear communication in the number of edges. We again use  $T$ , a threshold bounding the height of the fragments looking for outgoing edges.

To this end, we must find a way to detect the minimum outgoing edge in  $O(T/\epsilon)$  rounds. In fact, we want to get rid of the  $\log_{\log n} n$  (or  $\frac{\log n}{\log \log n}$ ) that appears in FINDMIN-C due to a  $\log n$ -ary search. As before, we assume that the maximum edge weight is less than  $n^k$  for some constant  $k$ . In KKT the range of edge weights is partitioned into  $\log n$  equal sized intervals; hence,  $\frac{\log n}{\log \log n}$  narrowing downs are needed to find the minimum outgoing edge. To speed up the process, we narrow down the interval by a factor of  $n^\epsilon$  each time, where  $n^\epsilon > \log n$ .

Similar to KKT, for the  $i^{\text{th}}$  interval we need one bit to be the result of TESTOUT( $x, I_i$ ), where  $I_i$  is the  $i^{\text{th}}$  part of the interval that we are searching. Then, the leader will pick the first part that has an outgoing edge, i.e. the first interval whose TESTOUT result is true, and again this new interval will be divided by  $n^\epsilon$ .

Nevertheless, in order to do this, each node needs an array of  $n^\epsilon$  bits. Sending an array with this magnitude will need  $\frac{n^\epsilon}{\log n}$  consecutive messages. Therefore, we use a *pipelining technique* so that every leaf sends these messages one after another without delay, and every internal node upon receiving the array from all its children calculates the sum of those arrays and sends it up. In fact, internal nodes will receive the parts of their children's arrays in order; therefore, they can compute the sum for the parts they have received (from

---

**Algorithm 1** Activation Algorithm

---

```

1: procedure ACTIVATE( $T$ ) ▷ Takes the value of threshold as input and activates the fragments w.r.t  $T$ .
2:   All of the leaves send up a timer with value  $T$ .
3:   Every internal node that received timers from all of its children sends up the minimum timer.
4:   Every leader that receives the timers from all of its children is activated.
5: end procedure

```

---

**Algorithm 2** Faster version of KKT when MST has low diameter

---

```

1: procedure DIAM-KKT
2:   Set  $\hat{D} = 1$ . ▷  $\hat{D}$  is the estimation of the MST's diameter.
3:   while  $\hat{D} \leq 2n$  do
4:     for  $i = 1$  to  $i < c \log n$  do ▷  $c$  is a sufficiently large constant.
5:       Call ACTIVATE( $\hat{D}$ ) to activate only fragments with height  $\leq \hat{D}$ .
6:       Active fragments find outgoing edges using FINDMIN-C within  $O(\hat{D})$  rounds.
7:     end for
8:     Call ACTIVATE( $\hat{D}$ ) ▷ Start of verification.
9:     Every active fragment uses HP-TESTOUT to decide w.h.p the existence of an outgoing edge.
10:    Every fragment whose HP-TESTOUT result is false broadcasts message STOP to all of its nodes.
11:    Every node that has not received STOP, restarts with  $\hat{D} = 2\hat{D}$ .
12:  end while
13: end procedure

```

---

all children) and send it up without waiting for the whole array to arrive.

It is easy to verify that the time rounds needed for the leader to receive the whole array from all its children is at most  $T + \frac{n^\epsilon}{\log n} \leq 2T$  if  $\epsilon < 1$ , because the height of the fragment cannot be more than  $T$ , otherwise it would not be allowed to look for outgoing edges.

Now we show how this pipelining technique helps in getting high probability in a single iteration. The idea is that instead of using only one hash function we use  $O(\log n)$  pipelined hash functions. But again, we do not wait for the results of the first hash function and then repeat the process with another one. We simultaneously (almost) apply  $\log n$  randomly chosen hash functions. The leader broadcasts all of the  $O(\log n)$  hash functions one after another and without any delay. This  $\log n$  extra messages through each link will not affect the time complexity because the extra  $O(\log n)$  time will be added to the threshold on fragment's height, which is relatively very large; this pipelining of hash functions only affects message complexity by a factor of  $O(\log n)$ . Algorithm FASTFINDMIN shows how finding the lightest outgoing edge can be done in  $O(T/\epsilon)$  with high probability using  $O(\log n)$  hash functions.

We also modify TESTOUT to take another argument  $h_j$  which is the hash function that it will use. Note that when we are in the first phase of the algorithm we can have  $O(\log n)$  iterations. Therefore, we will get high probability with applying only one hash function in FASTFINDMIN-C. We do not give the pseudocode separately for FASTFINDMIN-C. It is FASTFINDMIN but with one hash function instead of  $O(\log n)$ , and it gets constant probability of success.

**LEMMA 1.** *There exists a constant  $c$  for which, FASTFINDMIN finds the lightest outgoing edge w.h.p.*

**PROOF.** Imagine  $I$  is the current interval under search. Let  $I_f$  be the first subinterval that has an outgoing edge. Since, as stated in [KKT15], each  $h_i$  is an odd hash function, the probability that an odd number of outgoing edges in  $I_f$  hash into 1 is at least  $1/8$ . Note that non-outgoing edges will

cancel each other's parity since they appear exactly twice in the sum. Therefore, the probability of this event not happening is less than  $7/8$  for one hash function. Using  $c \log n$  hash functions this probability is reduced to  $\frac{7}{8}^{c \log n}$ . However, we need to narrow down the interval a total of  $k/\epsilon$  times, and the whole process fails if any of these narrowing downs fails. Hence, by union bound, the probability of not finding the lightest outgoing edge will be  $\frac{k}{\epsilon} \cdot (\frac{7}{8})^{c \log n}$ . Now, note that we always want  $n^\epsilon = \Omega(\log n)$ , because otherwise the whole array can be transmitted in one message; this implies that  $\epsilon > \Omega(\frac{\log \log n}{\log n})$ . Thus, for  $\frac{k}{\epsilon} \cdot (\frac{7}{8})^{c \log n}$  to be less than  $1/n^{c_1}$  it suffices that  $(\frac{7}{8})^{c \log n} < \frac{1}{n^{c_1+3}}$  which happens for  $c > \frac{c_1+3}{\log \frac{7}{8}}$ .  $\square$

Having all the prerequisites, we can provide the FASTMST algorithm which finds an MST in  $O(n/\epsilon)$  rounds and with  $O(\frac{n^{1+\epsilon} \log \log n}{\epsilon})$  communication w.h.p. FASTMST starts with setting the threshold value to  $\frac{n}{\log^2 n}$ . Before executing an iteration for finding outgoing edges the procedure ACTIVATE is called to make sure that only fragments whose height is below threshold are communicating and looking for outgoing edges. For every value of threshold (say  $T = \frac{n}{(\log^{(i)} n)^2}$ ),  $O(\log^{(i)} n)$  iterations are performed. Next theorem shows that this many iterations are enough to merge all fragments with height below  $T$ .

**THEOREM 4.** *Algorithm FASTMST terminates in  $O(n/\epsilon)$  rounds w.h.p.*

**PROOF.** We know from Lemma 1 that FASTFINDMIN fails probability  $1/n^{c_1}$  for some constant  $c_1$ . Now, in any iteration, there are at most  $n$  fragments. Therefore, by union bound, w.h.p the number of fragments is divided by at least 2 in each iteration. Hence, the number of iterations needed for each phase is log of the number of fragments. Furthermore, there are a total of  $\sum_{i=1}^{i=\log^* n} O(\log^{(i)} n) = O(\log n)$  iterations and they all succeed w.h.p, again by union bound. The maximum edge weight is  $n^k$  for some constant  $k$  and

---

**Algorithm 3** An algorithm for finding lightest outgoing edge in  $O(T)$  rounds w.h.p

---

```

1: procedure FASTFINDMIN( $x, \epsilon$ ) ▷ Takes fragment leader  $x$  and  $\epsilon$  as input.
2:   Initialize the search interval  $I$  to  $[1, n^k]$ .
3:   while size of the interval is more than 1 do
4:      $x$  broadcasts odd hash functions  $h_1, h_2, \dots, h_{c \log n}$  where  $h_j : [1, n^k] \rightarrow \{0, 1\}$ .
5:     Let  $I_i$  be the  $i^{th}$  part of the current interval. For  $i = 1$  to  $n^\epsilon$  and for  $j = 1$  to  $c \log n$  calculate TESTOUT( $x, I_i, h_j$ ) in the fragment leader by pipelining.
6:      $x$  determines, w.h.p, the first subinterval with an outgoing edge. This is done by picking the minimum  $i$  (say  $i_m$ ) for which there exists some hash function  $h_m$  such that TESTOUT( $x, i_m, h_m$ ) = true.
7:     Update  $I$  to be  $I_{i_m}$ , the first interval with an outgoing edge.
8:   end while
9:   Return the outgoing edge.
10: end procedure

```

---

**Algorithm 4** Linear time algorithm for finding the MST

---

```

1: procedure FASTMST( $\epsilon$ ) ▷ Takes  $\epsilon$  as input.
2:   Initialize  $F$ , set of all fragments, to be all of the singletons.
3:   Set threshold counter  $i = 1$ .
4:   while  $i \leq \log^* n$  do
5:     Set the threshold  $T = n / (\log^{(i)} n)^2$ .
6:     Set the iteration counter  $j = 1$ .
7:     while  $j \leq 2 \lceil \log^{(i)} n \rceil$  do
8:       Call ACTIVATE( $T$ )
9:       if  $i = 1$  then
10:        Call FASTFINDMIN-C( $x, \epsilon$ ) for every fragment leader  $x$ .
11:       else
12:        Call FASTFINDMIN( $x, \epsilon$ ) for every fragment leader  $x$ .
13:       end if
14:       Merge fragments using the lightest outgoing edges found.
15:       Increase  $j$  by one.
16:     end while
17:     Increase  $i$  by one.
18:   end while
19: end procedure

```

---

since each time we are narrowing down the range under search by  $n^\epsilon$  then a total of  $\frac{k}{\epsilon}$  narrowing downs are needed; hence the factor  $\frac{1}{\epsilon}$  in the time complexity.

When all of the fragments with height  $< n / (\log^{(i)} n)^2$  are merged, the number of remaining fragments cannot exceed  $(\log^{(i)} n)^2$ . Therefore, for each phase where the threshold is updated from  $n / (\log^{(i-1)} n)^2$  to  $n / (\log^{(i)} n)^2$ , we only need  $O(\log(\log^{(i-1)} n)^2) = O(\log^{(i)} n)$  iterations. Moreover, in the last phase the threshold is exactly  $n$  and all of the remaining fragments will merge.

In each iteration only fragments with height below the threshold are allowed to look for outgoing edges. Thus, the overall complexity will be

$$\frac{1}{\epsilon} \sum_{i=1}^{\log^* n} O(\log^{(i)} n) \cdot O\left(\frac{n}{(\log^{(i)} n)^2}\right) = \frac{n}{\epsilon} \sum_{i=1}^{\log^* n} O\left(\frac{1}{\log^{(i)} n}\right).$$

The last term of this sum is 1 and each time the denominator is losing a log; we can say the denominator is at least doubled each time. Therefore, this sum is less than  $\sum_{i=0}^{\infty} \frac{1}{2^i} = 1$ , and the theorem follows.  $\square$

**THEOREM 5.** Algorithm FASTMST requires  $O(\frac{n^{1+\epsilon} \log \log n}{\epsilon})$  messages.

**PROOF.** As stated in Theorem 4, it takes  $\frac{k}{\epsilon}$  narrowing downs before the lightest outgoing edge is found. For each

of these narrowing downs every node needs to communicate an  $n^\epsilon$ -bit array upward for each of the  $O(\log n)$  hash functions. This is  $O(n^\epsilon \log n)$  bits which can be pipelined in time  $O(n^\epsilon)$ . In total, every link will be used for  $\frac{k}{\epsilon} \cdot n^\epsilon$  messages after the first phase. Besides, there will be  $O(\log \log n)$  iterations over all phases  $i \geq 2$ . On the other hand, in the first phase, we use only one hash function but we have  $O(\log n)$  iterations. Thus, considering  $k$  is constant, the total message complexity will be:

$$O\left(\frac{1}{\epsilon} \frac{n^{1+\epsilon}}{\log n} \log n\right) + O\left(\frac{1}{\epsilon} \frac{n^{1+\epsilon}}{\log n} \log n \log \log n\right) = O\left(\frac{n^{1+\epsilon}}{\epsilon} \log \log n\right).$$

$\square$

## 5. LINEAR TIME ALGORITHM FOR SPANNING TREE

For the construction of spanning tree in linear time we again use a threshold  $T$  to only allow fragments with height less than  $T$  to look for outgoing edges. Like the previous section we need to boost the probability when we cannot have  $O(\log n)$  iterations. This will be done by running repetitions of each iteration in parallel.

However, the difficulty will come when we try to handle cycles. In ST-KKT a node realizes that it is on a cycle if it does not hear from at least one of its children after

enough time. More precisely, every leaf sends a message upward and every internal node waits to hear from all its children, and if it did not hear from them after enough time (say  $O(n)$  rounds) then it must have been on a cycle. In our algorithm, since we are using some threshold we cannot provide enough time to make sure that internal nodes hear from their children. In fact, if an internal node does not hear from all of its children it could be that it is on a cycle or the threshold was not large enough. We need to somehow distinguish between these two cases. In this section we will provide our algorithm for detecting and removing cycles. Throughout this section we assume that each edge has a unique ID made by concatenating the IDs of its endpoints, where the ID of the endpoint with smaller ID comes first.

## 5.1 Handling Cycles

After each fragment finds an outgoing edge, we need to break the cycles, i.e., remove an edge from the cycle. Let us define some important terms first. In iteration  $i + 1$  of the algorithm we have:

**Old fragment:** A fragment at the end of the  $i^{th}$  iteration.

**Marked edges of a node:** When an old fragment  $F$  finds an outgoing edge, the endpoint of the outgoing edge in  $F$  is called the tail, while the other endpoint is called the head. A node  $s$  can be the head of a number of outgoing edges found either on the this iteration or on previous iterations. These edges are considered the marked edges for node  $s$  unless they have been removed from the list of marked edges at some point during the algorithm. Note that an outgoing edge is not considered marked for its tail.

**Current leader (of an old fragment):** In iteration  $i + 1$  every old fragment tries to find an outgoing edge. If it succeeds in doing so, its current leader will become the tail of the outgoing edge. Otherwise, its current leader is the same as its leader at the end of iteration  $i$ .

**Cluster:** When finding outgoing edges is over, consider the graph formed by the old fragments and edges that are marked for some node at the moment. Each cluster is in fact a connected component in this graph. A cluster contains one or more of the old fragments at this iteration.

**Bad edge:** Is a marked edge which is either on a cycle (of any length) or on a path of length  $> T$  from a leaf in its cluster.

**1-bad and 2-bad fragments:** If an old fragment has 2 or more bad edges in total, it is called a 2-bad fragment. Note that the number of bad edges of an old fragment is the sum of bad edges of its nodes (including the current leader). A 1-bad fragment has exactly 1 bad edge.

Our goal is to keep the invariant that clusters with diameter less than  $T$  are merged into fragments with diameter greater than  $T$  before  $T$  is incremented.

The basic idea of removing cycles in time proportional to the threshold is to remove a cycle only when necessary. Later we will prove that a 2-bad fragment *has to* belong to a

cluster of diameter more than  $T$ . In fact, we will prove that the current leader of a 2-bad fragment has to be on a path of length more than  $T$  from a leaf in the cluster. Therefore, a cycle that lies in cluster containing 2-bad fragments can wait to be removed until the threshold gets big enough. However, we make sure that the cycle of a cluster with *only* 1-bad fragments is broken.

One problem is that we do not know if the bad edge of a 1-bad fragment is because of a cycle or a long path. Therefore, when we are eliminating edges to get rid of cycles we might be eliminating edges from a long path as well. This is problematic if the eliminated edges are in a cluster which has 2-bad fragments. In this case a 2-bad fragment might become a 1-bad fragment because we have removed an edge from a long path that was connected to it. We have to prevent this, otherwise new cycles of 1-bad fragments keep appearing that we have to deal with. In order to prevent a 2-bad fragment from becoming 1-bad, we eliminate an edge only if it is at distance more than  $T$  from the current leader of 2-bad fragments. In particular, 2-fragments signal 1-bad fragments that are within distance  $T$  and *forbid* them from eliminating edges.

In other words, we make sure that cycles in cluster that only have 1-bad fragments are broken, while we guarantee that eliminated edges do not result in more 1-bad fragments. The steps we follow to remove cycles are as follows:

- 1) Determining bad edges.
- 2) Current leader determines  $B$  the number of bad edges in its old fragment.
- 3) a) If  $B > 1$ , observe this occurs when there is a path of length more than  $T$  from a leaf to the current leader in the cluster. An edge is removed if it is at distance more than  $T$  from the current leader of a 2-bad fragment. 2-bad fragments send a *ChoiceForbidden* signal to 1-bad fragments within distance  $T$  and prevent them from eliminating edges.
- b) If  $B = 1$ , and the current leader has not received any *ChoiceForbidden* signal, it will choose to eliminate either the outgoing edge or the bad edge, depending on which has the smaller ID. Elimination occurs if an edge is chosen by both its endpoints for removal.

When cycles are broken, all fragments merge using the remaining outgoing edges. Note that there are still clusters which have cycles but these clusters have 2-bad fragments. Consequently, they will not look for outgoing edges until the threshold becomes large enough; hence, they cannot form new cycles. As we will prove later, a cluster cannot have more than one cycle. Therefore, the only cycles of such clusters will be removed with the same technique at higher thresholds.

Before presenting the algorithm for removing cycles we prove that clusters that have 2-bad fragments have to have diameter larger than  $T$ ; hence, they can be dealt with in higher thresholds. The following lemma proves this.

**LEMMA 2.** *If an old fragment  $F$  is 2-bad the diameter of its cluster is more than  $T$ .*

**PROOF.** First we observe that if  $F$  is on a cycle it has to have an outgoing edge. The reason is that every fragment can have at most one outgoing edge; hence, the fragments connected to  $F$  will form a tree like structure unless  $F$  itself has an outgoing edge to one the fragments connected (di-

rectly or indirectly) to it. Now, for the same reason all of these connected fragments cannot have more than one cycle because if they did one of the fragments should have had more than one outgoing edge which is a contradiction. Thus, if we imagine that  $F$  has two or more bad incoming edges at most one of them could be due to a cycle and the other must be the result of a long height (compared to threshold) that has prevented some timer to reach high enough. This ensures the existence of path with length more than  $T$  in  $F$ 's cluster and the lemma follows.  $\square$

Algorithm CYCLEREMOVER shows how to break cycles. When fragments find their outgoing edges, we make sure that cycles in clusters with only 1-bad fragments are removed. We do this carefully so that no 2-bad fragment becomes 1-bad after some edges are eliminated. We assume every node keeps a list of marked edges. All nodes only add to their list of marked edges over the course of algorithm unless they receive a *Clear* message from the leader which tells them to empty their lists. A node will receive the *Clear* message when it is no longer on a cycle and the height of its cluster is no more than  $T$ .

In CYCLEREMOVER, we use timers as in the MST algorithm. Every leaf sends up a timer with value  $T$  upward to the current leader. Every internal node (including the current leader) that receives the timer from all of its children sends up the timer with minimum value. As a result, after  $T$  rounds every node will know which of its marked edges it has not heard from, i.e. bad edges. Then every node of an old fragment sends up the number of its bad edges to the current leader. Again after  $T$  rounds, the current leader will know how many bad edges its fragment has; hence, 1-bad and 2-bad fragments are determined. Now, before we break the cycles, we make sure that edges that are within distance  $T$  of a 2-bad fragment are not eliminated. To this end, the current leader of every 2-bad fragment sends down the *ChoiceForbidden* message which is attached to a timer with value  $T$ . The leader of any 1-bad fragment that receives such a message does not eliminate any edges. In order to prevent redundancy, if a *ChoiceForbidden* messages reaches the leader of some 2-bad fragment, it will not be forwarded because that leader has already sent down one such message.

1-bad fragments that have not received the *ChoiceForbidden* message eliminate either the outgoing edge or the bad edge. In particular, the fragment picks edge that has the smaller ID for removal. This edge is removed only if the fragment on the other side picks it for removal, as well. The next lemma shows that eliminating the node with smaller ID will break the cycles, and this does not affect the asymptotic number of iterations we need to merge all the fragments with height below the threshold.

**LEMMA 3.** *If all of 1-bad fragments that have not received the *ChoiceForbidden* message pick between their bad edge and outgoing edge the one with smaller ID and edges picked by both endpoints are removed, cycles in cluster with only 1-bad fragments will break. This does not affect the asymptotic number of iterations required for merging fragments in the current phase.*

**PROOF.** Consider a cycle  $C$  that is not broken after calling CYCLEREMOVER. If there is at least one 2-bad fragment in that cycle, that fragment will broadcast the *ChoiceForbidden* message and prevents the height of the cluster from

becoming less than  $T$ . This whole cluster along with the cycle will be dealt with in higher thresholds. Now, if there is no 2-bad fragment in the cycle, having a cycle itself implies that all of the fragments have to be 1-bad. These 1-bad fragments will not receive any *ChoiceForbidden* message and hence choose between their outgoing edge and bad edge. On a cycle, the edge with minimum ID among all of the outgoing and bad edges is certain to be eliminated by both of the old fragments connected to it, since they both agree on its elimination. As a result, the cycle will break.

Recall that a 1-bad fragment is either because of cycle or a long path. Let us say the number of 1-bad fragments that have not received *ChoiceForbidden* on cycle or on top of long path is  $k$ . Since only edges with smaller ID are removed after the agreement on both sides, the number of removed edges by CYCLEREMOVER is at most  $k/2$ . Therefore, the asymptotic number of iterations needed for merging fragments still remains the same.  $\square$

The algorithm FASTST is shown for finding the spanning tree in  $O(n)$  rounds and using  $O(n \log n \log \log n)$  messages. We do not give the pseudocode for LOGFINDANY. It does the same thing as FINDANY-C except it uses  $O(\log n)$  hash functions with the same pipelining technique to get high probability of success in finding outgoing edges.

Over the course of the algorithm, we use  $\log^* n$  phases and in each phase we have the threshold  $T = n / (\log^{(i)} n)^2$ . In any iteration FINDANY-C and LOGFINDANY spend time proportional to the height of the fragment which is bounded by  $T$ . Note that using  $O(\log n)$  hash functions in a pipelined manner in LOGFINDANY does not affect time complexity. Therefore, similar to the analysis of Algorithm 4, the time complexity is  $O(n)$ . Moreover, we are using  $O(\log n)$  hash functions after the first phase. As in Theorem 5, the overall number of iterations after the phase is  $O(\log \log n)$ . Hence, the message complexity is  $O(n \log n)$  for the first phase, and  $O(n \log n \log \log n)$  for the rest of the phases which is  $O(n \log n \log \log n)$  overall.

## 6. CONCLUSION

We have shown that time can be brought down to linear in  $n$  while maintaining a communication which is sublinear in  $m$  when the average degree is  $\omega(n^\epsilon)$  for the MST and  $\omega(\log n \log \log n)$  for the spanning tree. We have also shown the first minimum spanning tree construction algorithm which uses time approximately proportional to the diameter of the MST without paying an excessive cost for communication.

Some intriguing questions remain: Can time be reduced to  $o(n)$  when the network has small diameter, but the diameter of the MST is large, without  $m$  communication? Can MST be computed with a deterministic algorithm, or a Las Vegas when nodes do not know the exact size of the network, or in an asynchronous model, with  $o(m)$  communication?

## 7. REFERENCES

- [Awe87] Baruch Awerbuch. Optimal distributed algorithms for minimum weight spanning tree, counting, leader election, and related problems. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 230–240. ACM, 1987.

---

**Algorithm 5** The algorithm for removing cycles

---

```
1: procedure CYCLEREMOVER( $T$ ) ▷ Takes the threshold as input.
2:   All leaves send up a timer with value  $T$ , and internal nodes sends up the minimum timer of their children.
3:   After  $T$  rounds every node determines its bad edges.
4:   Every node sends the number of its bad edges to the current leader of the old fragment.
5:   After  $T$  rounds every current leader will know if its fragment is 2-bad or 1-bad.
6:   Current leader of 2-bad fragments send down a ChoiceForbidden message attached to a timer.
7:   1-bad fragments who did not receive the ChoiceForbidden message are allowed to eliminate either the bad or the outgoing edge. They eliminate the edge with smaller ID only if the fragment on the other side does not want to keep it.
8: end procedure
```

---

**Algorithm 6** Linear time algorithm for finding ST

---

```
1: procedure FASTST
2:   Initialize  $F$ , set of all fragments, to be all of singletons.
3:   Set threshold counter  $i = 1$ .
4:   while  $i \leq \log^* n$  do
5:     Set the threshold  $T = n / (\log^{(i)} n)^2$ .
6:     Set the iteration counter  $j = 1$ .
7:     while  $j \leq 2\lceil\log^{(i)} n\rceil$  do
8:       Call ACTIVATE( $T$ )
9:       Active leaders broadcast the Clear to make all nodes empty their list of marked edges.
10:      if  $i = 1$  then
11:        Call FINDANY-C( $x$ ) for every active fragment leader  $x$ .
12:      else
13:        Call LOGFINDANY( $x$ ) for every active fragment leader  $x$ .
14:      end if
15:      Call CYCLEREMOVER( $T$ )
16:      Merge fragments using the outgoing edges that were not removed in CYCLEREMOVER.
17:      Increase  $j$  by one.
18:    end while
19:    Increase  $i$  by one.
20:  end while
21: end procedure
```

---

- |         |   |                       |   |
|---------|---|-----------------------|---|
| [CT85]  | F Chin and HF Ting. An almost linear time and $O(n \log n + e)$ messages distributed algorithm for minimum-weight spanning trees. In <i>Foundations of Computer Science, 1985., 26th Annual Symposium on</i> , pages 257–266. IEEE, 1985.             | [GKP98]               | Juan A Garay, Shay Kutten, and David Peleg. A sublinear time distributed algorithm for minimum-weight spanning trees. <i>SIAM Journal on Computing</i> , 27(1):302–316, 1998.   |
| [Elk04] | Michael Elkin. A faster distributed protocol for constructing a minimum spanning tree. In <i>Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms</i> , pages 359–368. Society for Industrial and Applied Mathematics, 2004. | [GP16]                | Mohsen Ghaffari and Merav Parter. Mst in log-star rounds of congested clique. In <i>Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing</i> , PODC ’16, pages 19–28, New York, NY, USA, 2016. ACM.   |
| [Elk06] | Michael Elkin. An unconditional lower bound on the time-approximation trade-off for the distributed minimum spanning tree problem. <i>SIAM Journal on Computing</i> , 36(2):433–456, 2006.  | [HPP <sup>+</sup> 15] | James W Hegeman, Gopal Pandurangan, Sriram V Pemmaraju, Vivek B Sardeshmukh, and Michele Scquizzato. Toward optimal bounds in the congested clique: Graph connectivity and mst. In <i>Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing</i> , pages 91–100. ACM, 2015. |
| [Gaf85] | Eli Gafni. Improvements in the time complexity of two message-optimal election algorithms. In <i>Proceedings of the fourth annual ACM symposium on Principles of distributed computing</i> , pages 175–185. ACM, 1985.                                | [KKT15]               | Valerie King, Shay Kutten, and Mikkel Thorup. Construction and impromptu repair of an mst in a distributed network with $O(m)$ communication. In <i>Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing</i> , pages 71–80. ACM, 2015.                                    |
| [GHS83] | Robert G. Gallager, Pierre A. Humblet, and Philip M. Spira. A distributed algorithm for minimum-weight spanning trees. <i>ACM Transactions on Programming Languages and systems (TOPLAS)</i> , 5(1):66–77, 1983.                                      | [KP95]                | Shay Kutten and David Peleg. Fast distributed construction of k-dominating sets and applications. In <i>Proceedings of the fourteenth</i>   |

- annual ACM symposium on Principles of distributed computing*, pages 238–251. ACM, 1995.
- [KPP<sup>+</sup>15] Shay Kutten, Gopal Pandurangan, David Peleg, Peter Robinson, and Amitabh Trehan. On the complexity of universal leader election. *Journal of the ACM (JACM)*, 62(1):7, 2015.
  - [LPSPP05] Zvi Lotker, Boaz Patt-Shamir, Elan Pavlov, and David Peleg. Minimum-weight spanning tree construction in  $O(\log \log n)$  communication rounds. *SIAM Journal on Computing*, 35(1):120–131, 2005.
  - [Pel00] David Peleg. Distributed computing. *SIAM Monographs on discrete mathematics and applications*, 5, 2000.
  - [PRS16] Gopal Pandurangan, Peter Robinson, and Michele Scquizzato. A time-and message-optimal distributed algorithm for minimum spanning trees. *arXiv preprint arXiv:1607.06883*, 2016.
  - [SHK<sup>+</sup>12] Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. Distributed verification and hardness of distributed approximation. *SIAM Journal on Computing*, 41(5):1235–1265, 2012.

# Lock-free Linearizable 1-Dimensional Range Queries

Bapi Chatterjee<sup>\*</sup>  
 Chalmers University of Technology, Gothenburg, Sweden  
 bapic@chalmers.se

## ABSTRACT

Efficient concurrent data structures that support range queries are highly sought-after in a number of application areas. For example, the contemporary big-data processing platforms employ them as in-memory index structures for fast and scalable real-time updates and analytics, where analytics utilizes the range queries.

In this paper, we present a generic algorithm to perform linearizable range queries in lock-free ordered 1-dimensional data structures. The algorithm requires single-word atomic compare-and-swap (**CAS**) primitives. Our method generalizes the lock-free data structure snapshot of Petrank et al. [25]. Fundamentally, we utilize a partial snapshot object derived from the snapshot object of Jayanti [20].

We experimentally evaluate the proposed algorithm in a lock-free linked-list, skip-list and binary search tree (BST). The experiments demonstrate that our algorithm is scalable even in the presence of high percentage of concurrent modify operations and outperforms an existing range search algorithm in lock-free k-ary trees in several scenarios.

## Keywords

range query, range search, linearizability, concurrency, lock-free, data structure

## 1. INTRODUCTION

An ordered set abstract data type (ADT), which supports set operations - **Add**, **Remove**, **Contains** and **RangeSearch**, is a widely used programming interface. Sequential data structures such as linked-lists, skip-lists and BSTs, which provide predecessor queries, implement this ADT. With the ubiquity of multi-core processors, efficient concurrent version of these data structures are ever more important.

---

\*This work was partially supported by the Swedish Foundation for Strategic Research as part of the project RIT-10-0033 “Software Abstractions for Heterogeneous Multi-core Computer”.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ICDCN '17, January 04 - 07, 2017, Hyderabad, India

© 2017 Copyright held by the owner/authors. Publication rights licensed to ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007771>

An extremely important application area for such concurrent data structures is a modern big-data processing platform such as Google’s Bigtable [7] and Apache HBase [15]. These technologies, in addition to the on-disc storage for persistence, employ a concurrent data structure as an in-memory index for fast real-time data updates and analytics. Specifically, the analytics components thereof are based on the **RangeSearch** operations. Therefore, to ensure the correctness of the analytics, the consistency of a **RangeSearch**, in presence of other concurrent set operations, becomes paramount. Additionally, the overall performance of such applications gets significant impact from the scalability of the utilized concurrent index structure.

The most common consistency framework used in concurrent settings is linearizability [17], which makes a user perceive an operation to take effect instantaneously at a point between the invocation and response of the operation. In effect, linearizability of range search is highly desired because it ensures that the output provides an “aligned view”, with respect to a global real-time order, of the targeted entries in the data structure.

Traditionally, concurrent data structures use mutual exclusion locks for consistent set operations. However, locks succumb to high contention and often offer poor scalability. In addition to that, in an asynchronous shared-memory system, where an infinite delay or a crash failure of a process is possible, a lock-based concurrent data structure is vulnerable to pitfalls such as deadlock, priority inversion and convoying. On the other hand, in a lock-free data structure, processes do not hold locks and at least one non-faulty process is guaranteed to finish its operation in a finite number of steps. Therefore, lock-free data structures foster both scalability and fault tolerance.

A number of lock-free ordered data structures exist in the literature: singly-linked lists [13,14], doubly-linked lists [30], skip lists [21,29], BSTs [9,11,12,18,23], etc. However, in general, the ADT implemented by these data structures do not support **RangeSearch** operations. A linearizable **RangeSearch** outputs a consistent view of the concurrent data structure with respect to multiple points stored in it and thus is inherently different from other set operations that essentially require single point queries.

This paper makes the following contributions:

1. We present a generic method to implement linearizable **RangeSearch** operations in a lock-free ordered search data structure that supports linearizable **Add**, **Remove** and **Contains** operations.
2. Our method can be seamlessly integrated to any lock-free

- 1-dimensional ordered data structure.
3. We experimentally show that the proposed range search algorithm achieves good scalability and outperforms an existing range search method [6] in high contention scenarios.

## 1.1 Related work

To our knowledge, the only work existing in the literature with regard to a linearizable range query in a lock-free data structure is by Brown et al. [6], who implemented it in the lock-free k-ary search trees. Their method requires range scanning using a depth-first-search followed by validating the scan, and if any node is found outdated, the range scan is retried, often repeatedly. Thus, in the cases of multiple concurrent updates, this method lets the range search starve.

The **ConcurrentSkipListSet** in Java concurrency library supports **subSet** operations that are effectively non-linearizable range queries. Avni et al. [4] used software transactional memory (STM) to encapsulate the linearizable range queries in a lock-based concurrent skip list, which they call a Leap list. The scalability of their method mainly depends on the efficiency of the underlying STM. Moreover, an STM based approach for a range query incurs increasing overhead with the growth in the number of target shared-memory words covered by a transaction and thus may not be an efficient approach to perform queries for long ranges in a reasonably populated concurrent data structure. Sagonas et al. [27], proposed to implement linearizable range queries by way of locking each of the nodes, which contain the points in the target range, in a lock-based data structure. Recently, Basin et al. [5] published a brief announcement of a concurrent key-value store that supports range queries. However, we could not find a full description of their work.

## 1.2 A summary of our work

### 1.2.1 Background

A simple approach to perform a linearizable range search in a lock-free data structure is to collect a lock-free linearizable snapshot of the data structure and output the appropriate subset thereof. However, not many lock-free data structures support linearizable snapshot collection. Exception is - lock-free hash trie by Prokopec et al. [26]. They used a variant of double-compare-single-swap (DCSS) primitive, which is implemented using single-word CAS. Here, an update with a concurrent snapshot requires copying each updated node together with the path from the root to it to help the snapshot collection, which results in a performance deterioration of concurrent updates with snapshot. Also, the hash mapping of the keys makes it hard to compute a range search directly from a collected snapshot.

Petrrank and Timmat [25] presented another approach for collecting a linearizable snapshot of lock-free linked lists and skip lists. Their method is based on the wait-free multi-writer multi-scanner *snapshot object* of Jayanti [20]. A snapshot object, which consists of multiple shared-memory words, supports concurrent **scan** and **update** methods. An **update** writes a new value at one of the words, and a **scan** returns an atomic view of all the words.

Nonetheless, directly using a “full” snapshot collection for a **RangeSearch** is undesirable as it discards the advantage of the size of a range query output, which may not necessarily be equal to the size of the entire data structure.

A way to align the snapshot collection with the size of the queried range can be derived from the notion of the *partial snapshot* introduced by Attiya et al. [3]. A partial snapshot collection is essentially a generalization of the snapshot collection, in which the target is restricted to only a part of the multi-word object. This work underlines the motivation that a partial snapshot collection must be cheaper than a (full) snapshot collection.

The fundamental idea of a partial snapshot is based on the multiple concurrent announcements of *partial scan* operations. To handle the announcements, Attiya et al. used the idea of an *active set object* of Afek et al. [2]. An active set provides the methods **- join**, **leave** and **get\_set**. A **join** method is called to join the set of “active” processes, a **leave** is called to leave such a set, and **get\_set** outputs the list of processes which are part of the active set. Thus, an active set implementation keeps track of the processes that “currently” perform a partial **scan**.

### 1.2.2 Our approach

To implement a linearizable range search, we combine the method of lock-free data structure snapshot of [25] with an implementation of an active set.

The key element of the lock-free data structure snapshot of [25] is a special object called *snap-collector*. The data structure is augmented with a pointer to snap-collector allocated by a process (called a scanner) collecting a snapshot. Concurrent scanners use a single snap-collector to return the same snapshot. An operation such as **Add**, **Remove** or **Contains** *reports* a modification to a concurrent scanner using its snap-collector.

We modify a snap-collector by equipping it with the lower and upper limits of a target range and call the modified object a *range-collector*. However, it does not provide a scalable solution for a common scenario in which concurrent range queries target disjoint ranges and thus can not use the same range-collector. Naturally, to solve that issue, we need multiple “simultaneously active” range-collectors corresponding to the concurrent range queries targeting disjoint ranges. Yet, there can be scenario where the concurrent range queries have overlapping ranges and for them ensuring linearizability needs extra care. We explain it below.

Suppose  $op_1$  and  $op_2$  are two concurrent **RangeSearch** operations with overlapping but non-identical target ranges. Now, if  $op_1$  has its linearization point before that of  $op_2$ , and it includes a data-point  $k$  in its output, which belongs to the intersection of the associated target ranges, and there is no linearization point of a **Remove( $k$ )** between the linearization points of  $op_1$  and  $op_2$ ,  $op_2$  must also include  $k$  in its output for consistency. That effectively means that  $op_1$  and  $op_2$  must synchronize before their returns. Furthermore, if for concurrent and independent progress we use different range-collectors for  $op_1$  and  $op_2$ , a concurrent **Add/ Remove/ Contains** operation, say  $op_3$ , will have to report to both the range-collectors, if its return point happens to belong to the intersection of the two associated ranges. Naturally, it significantly increases the overhead of reporting, which is undesirable.

To overcome these issues, we augment a lock-free concurrent data structure with a lock-free linked-list of range-collector objects. Functionally, the linked-list implements an active set of processes performing concurrent range queries. The data structure holds a pointer to one of the ends of this

list, say the *head*. A new range-collector is allowed to be added only at the other end, say the *tail*. Thus, the addition to, removal from and traversal through this lock-free list delegate the active set methods `join`, `leave` and `get_set`, respectively.

A `RangeSearch` operation starts with traversing through the list and checks whether there is an *active* range-collector with an identical target range. If in the list such an active range-collector is found, it is used for a concurrent *coordinated* range scan, which is analogous to using same snap-collector by multiple concurrent scanners in [25]. A range-collector is removed from the lock-free linked-list as soon as the range scan at it gets over. With that, if concurrent `RangeSearch` operations  $op_1$  and  $op_2$  have overlapping ranges, and  $op_1$  successfully joins the active set before  $op_2$  by adding its range-collector to the lock-free list, we first make  $op_2$  help  $op_1$  to finish, and then let it restart. Thus at any time-point, the augmented list contains active range-collectors with disjoint ranges only.

It is easy to see that in our method a `RangeSearch` does not require multiple (repeated) scans followed by validation, and thus, presents a scalable method even in presence of high percentage of concurrent `Add`/ `Remove` operations in the data structure.

Please note that, similar to [25], our approach to implement range search in a lock-free data-structure is independent of the actual data structure design, and thus is generic.

Moreover, this work also practically presents an alternate method to implement a partial snapshot object derived from [20]. The existing algorithms for partial snapshot by Attiya et al. [3] and Imbs et al. [19] are based on repeated scan and validation that we aimed to avoid.

Lastly, because the range search in data structures storing multidimensional points is not similar to that in 1-dimensional data structures, for detail see Samet’s book [28], we do not claim that our method can be directly adapted to multi-dimensional range search problems in a concurrent setting, which in its own merit is a very important but yet largely unexplored topic.

In the remaining of the paper, we describe the algorithm in section 2. In section 3, we present a proof of the algorithm. The section 4 presents details of the experiments and discussion on the observations thereof. Section 5 concludes the paper.

## 2. THE LOCK-FREE RANGE SEARCH

We consider an *asynchronous shared memory system*  $\mathcal{U}$  which comprises a set of word-sized *objects*  $\mathcal{V}$  and a finite set of processes  $\mathcal{P}$  and supports (operations) *primitives* `read`, `write` and `CAS` (compare-and-swap).  $\mathcal{U}$  guarantees that the primitives are *atomic* i.e. they take effect instantaneously at an indivisible time-point [16]. Each object  $v \in \mathcal{V}$  has a unique *address*, commonly known as a *pointer* to  $v$ , denoted by  $v.ref$ .  $\text{CAS}(v.ref, exp, new)$  compares the value of  $v$  with  $exp$  and on a match updates it to  $new$  in a single atomic step and returns `true`; else it returns `false` without any update at  $v$ . Let  $|\mathcal{P}|=n$ . Processes  $p_i \in \mathcal{P}$  communicate by accessing the objects  $v \in \mathcal{V}$  using a primitive.

We consider an abstract data type (ADT) OSet that provides definition of operations `Add`, `Remove`, `Contains` and `RangeSearch` (*range search*). We expect a reader to be familiar with these operations as in their most common forms. The ADT OSet is implemented by an ordered 1-dimensional

data structure.

Fundamentally, our algorithm presents a generalization of the lock-free iterator algorithm of [25]. Therefore, for the sake of completeness, we briefly recap the algorithm of [25] in the section 2.1 before describing the linearizable range query algorithm in the section 2.2.

### 2.1 Snap-collector implementation

The lock-free data-structures considered in [25] are - linked list [14] and skip list [21], in which each node  $x$  has a unique key  $key(x)$  and it is straightforward to find  $next(x)$ . The data-structures implement an ADT that provides the operations `Add( $x$ )`, `Remove( $x$ )` and `Contains( $x$ )`. Without any ambiguity, we denote `Add(key( $x$ ))` by `Add( $x$ )`. In these data-structures, the operation `Remove( $x$ )` takes more than one atomic `CAS` steps. One of the `CAS` steps, whose success ensures that  $x$  will be eventually removed, is generally known as *logical remove*, and is considered the linearization point of `Remove( $x$ )`.

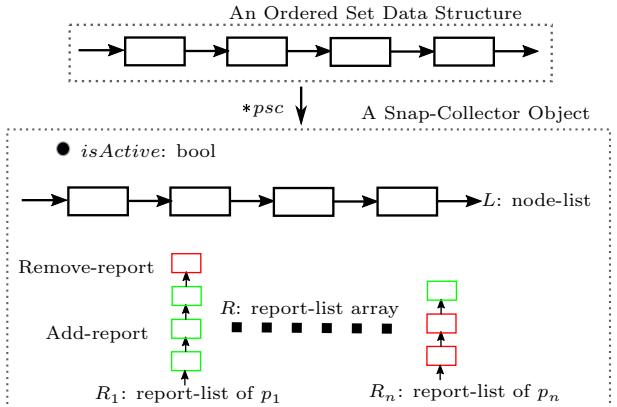


Figure 1: A snap-collector implementation

To implement the snapshot collection, a pointer  $psc$  to an instance of a class *snapshot* is maintained in the data-structure. The structure of the class *snapshot* is shown in the fig. 1. It holds (a) a boolean variable *isActive* (b) a list  $L$  of the data structure node pointers in which nodes are sorted by their keys and (c) an array  $R$  of lists of *reports*. A report comprises a node pointer and a *report-type* (ADD or REM). The size of the report-list array  $R$  is equal to the number of processes in the system. A *snapshot* with false *isActive* field is called *inactive*, otherwise *active*. If there is no ongoing snapshot collection, the pointer  $psc$  holds a reference to an inactive *snapshot*. Initially  $psc$  points to a dummy *snapshot* which is always inactive.

A snapshot collection starts with reading the pointer  $psc$ . If  $psc$  is found pointing to an inactive *snapshot*, a new active *snapshot* is allocated. The *snapshot* attempts to update  $psc$  to point to the new *snapshot* using a `CAS`. If the `CAS` fails, it helps the concurrent operation that successfully injected its own *snapshot*. In both the situations (success in injecting own *snapshot* or helping), the process scans the data-structure and collects the address of every node  $x$  that it discovers which is not logically removed. The list  $L$  (which has very similar semantics as the lock-free queue of Micheal et al. [22]) is used for the purpose in which the node  $tail(L)$  satisfies  $key(tail(L)) \leq key(node)$  for every node  $node \in L$ .

---

```

data-structure-node.
1 struct DSNode {K k; ...;} *DSNPtr;
  report consisting a DSNPtr and report-type.
2 struct Report {RptType: ADD, REM.
3   RptType rt; DSNPtr node;
4 } *RptPtr;
  packet of a DSNPtr and auxilliary info to find
  next node.
5 struct Window {DSNPtr cur;...;};
6 struct RCNode {range-collector-node.
7   K lo, hi; Window * L;
8   RptPtr * R; Array-Size=#Threads.
  mark bit of next is used.
9   (RCNPtr ref, bool mark) next;
10 } *RCNPtr;
  Global shared variables to initialize.
11 RTail = RCNode(∞, ∞, null, null, (null, 1));
12 RHead = RCNode(-∞, -∞, null, null, (*RTail, 1));
13 ...;Other global variables.
  Returns the Window of DSNPtr with key just ≥
  lo; returns null if no such node present.
14 FindMinNode(K lo)
  Returns a Window of DSNPtr with key just ≥ key(x
  →cur).
15 Next(Window x)
  Returns true if x is not logically removed.
16 IsPresent(DSNPtr x)
  Returns true if range-collection at rn is not
  linearized i.e. its next field's mark is 0.
17 IsActive(RCNode rn)

```

---

Returns the Window of DSNPtr with largest key in  $rn \cdot L$  after attempting to add  $x$  to it.

18 AddNode(RCNode rn, Window x)

Adds a Window of DSNPtr with max-value key to  $rn \cdot L$  and sets 1 at the mark of  $rn$ 's *next*.

19 Deactivate(RCNode rn)

20 | AddNode(rn, Window(\*DSNNode(∞),...));
 | (ref, m) = rn.next;
 | while m≠1 or !CAS(rn.next, (ref, 0), (ref, 1)) do
 | | (ref, m) = rn.next;

Atomically adds a dummy report to each of  $A[tid]$  and  $R[tid]$  lists of  $rn$ .

24 BlockFurtherReports(RCNode rn)

Returns whether the interval  $[lo, hi]$  isContained/contains/overlaps/isDisjoint with the interval of the range-collector-node  $rn$ .

25 ChkOverlap(K lo, K hi, RCNode rn)

Returns a list of keys in interval  $[lo, hi]$  from the range-collector-node  $rn$ .

26 ProcessRange(K lo, K hi, RCNode rn)

Returns true if range-collection at  $rn$  is not linearized i.e. its *next* field's *mark* is 0.

27 Report(RCNode rn, Report r, int tid)

---

Collects the nodes in  $[lo, hi]$  to add to  $rn \cdot L$ .

28 SnapRange(K lo, K hi, RCNode rn)

29 | w = FindMinNode(lo);
 30 | if w.cur≠null then
 31 | | while IsActive(rn) and w.cur.key≤hi do
 32 | | | if IsPresent(w.cur) then w = Next(AddNode(rn, w));
 33 | | | Deactivate(rn); BlockFurtherReports(rn);

---

**Algorithm 1.** The basic methods of the range query algorithm

To optimize the scanning by multiple concurrent processes, a process is allowed to insert a node-pointer  $*x$  in  $L$  only if  $key(x) > key(tail(L))$ . An attempt to insert  $*x$  in  $L$  returns  $tail(L)$  if  $key(x) \leq key(tail(L))$  (without making any changes in  $L$ ) and  $*x$  is returned if it was successfully added to  $L$ . The *isActive* variable is checked before every attempt to read a node from the data-structure and if it is found false then the snapshot collection is guaranteed to have finished. On completion of scanning the data-structure, *isActive* is set false. This is the linearization point of the snapshot. Further, in order to ensure that all the processes have the same view of  $L$ , before attempting to set false at *isActive*, a null is inserted at  $L$ .

For a linearizable snapshot collection, a process performing Add/ Remove/ Contains requires to report the operation. A Remove( $x$ ), on finding  $x$ , performs steps up to the logical remove of  $x$ , then before taking further steps, adds a REM report consisting  $*x$  to its respective report-list in  $R$ . An Add( $x$ ) after adding  $x$  with a successful CAS, reports  $*x$  as an ADD to its respective report-list in an active snap-collector. Before every Add report, the logically remove status of  $*x$  is checked to avoid an unnecessary reporting. A Contains( $x$ ) on finding a node reports (ADD or REM) to an active snap-collector. To ensure a consistent view of the report-lists of all the processes in the system, before the *isActive* field of a snap-collector is set false, a null report is added to each of the report-lists. After the linearization of the snapshot, the reports are processed by sorting them and then merging them to  $L$ . A processed snapshot contains addresses to the nodes present in the data structure during the execution interval of the snapshot collection. The concurrent processes working at the same snap-collector return same snapshot.

## 2.2 Lock-free linearizable range search algorithm

Having presented the basic construction of the snapshot collection method, we are now prepared to present the linearizable range search by means of an active set of processes that collect concurrent snapshots targeting different subsets of nodes present in a lock-free data-structure.

In the description of the algorithm, we assume that the memory allocator always allocates a new shared-memory object at a new address and thus ABA<sup>\*</sup> problem never occurs. We also assume that the memory reclamation is lock-free such as one using a lock-free garbage collector, for example, [24]. These are standard assumptions in the description of a lock-free data structure algorithm.

The pseudo-code of the algorithm is given in the algorithms 1 to 4. We have sufficiently commented the pseudo-code to describe the functionality of the methods.

We consider an ordered lock-free data-structure with nodes of type DSNode which are indexed by the keys  $k$  uniquely selected from a partially ordered universe. As before, we shall use  $k$  to denote a node if its key is  $k$ .

The data-structure is augmented with a lock-free linked list of instances of the class RCNode (range-collector node), see line 6 to line 10 in the algorithm 1. An RCNode, similar to a snap-collector node, in addition to containing the data structure node list  $L$  and array of report-lists  $R$ , contains a pointer *next* to connect to another RCNode in the list.

<sup>\*</sup>ABA is a short-cut for stating that a value at a shared variable can change from A to B and then back to A, which is a fundamental problem in a CAS-based lock-free algorithm, and if not remedied, it can corrupt the semantics of the algorithm.

It also records the range in terms of the lower limit  $lo$  and the upper limit  $hi$ . However, unlike a snap-collector node, a range-collector node does not contain the boolean field *isActive* to fix the linearization point. Here we use a bit from the pointer *next* for the purpose, which is a design optimization.

An important difference between an RCNode and a snap-collector node of [25] is the list  $L$ : we store additional info in a node (of type **Window**, line 5) of  $L$  that can facilitate traversal in the data-structures like BST, in which, unlike linked list and skip list, computing  $next(x)$  is not straightforward. We describe in section 2.3 how we implemented the method *Next* that finds the successor of a node in a BST.

The list of RCNodes, which is unordered, has similar semantics as the lock-free linked list of Harris [14], except that here a new RCNode can be added only at one end. See fig. 2. The linked list is represented by two sentinel RCNodes  $RHead$  and  $RTail$  with ranges  $[-\infty, -\infty]$  and  $[\infty, \infty]$ , respectively, s.t. they are disjoint to any RCNode ( $-\infty$  and  $\infty$  are the minimum and the maximum elements, respectively, in the partially ordered universe of keys). At the initialization of the data-structure, the *next* of  $RHead$  points to  $RTail$  and  $L$  and  $R$  of both these RCNodes are null. Additionally, the *next* of  $RTail$  is always null.

A new RCNode is added using a **CAS** such that its *next* pointer always points to  $RTail$ . On a **CAS** failure we restart from  $RHead$ . To remove an RCNode from the linked list, first its *next* pointer is marked (one unused bit is set using a **CAS**) and then the *next* pointer of the previous RCNode is updated. Any traversal through the list always helps a pending remove in the path. The method *RangeSearch* of the data-structure (see line 34 to line 57 in algorithm 2) intrinsically uses add and remove of an RCNode in the unordered list together with the traversal through the list.

In effect, at any point of time, all the processes that perform *RangeSearch* use exactly one RCNode whose *next* is not marked. A traversal through the list outputs the process-id of all the processes that perform *RangeSearch*. Because a new RCNode is added only at the  $RTail$  (just before it), a traversal can not miss a new *RangeSearch* that started before and has not linearized yet. Therefore, a process that starts a new *RangeSearch* always gets to know the concurrent processes that are active and performing *RangeSearch*. Also, “logically” leaving this set of active processes takes one **CAS** execution. Thus, the augmented lock-free list implements an active set.

A process intending to perform *RangeSearch*( $[lo hi]$ ), line 34 to line 57 in algorithm 2, starts scanning the unordered list starting from  $RHead$ . A variable *mode* is assigned value **INIT** to indicate that the process is yet to start the collection of desired data-structure nodes, line 35. On reading an RCNode *cur* with range  $[x y]$ , one of the following is done depending on how  $[lo hi]$  relates to  $[x y]$ :

1. If the *next* of *cur* is marked and not **null** (line 39), it indicates a pending but linearized *RangeSearch* operation. We help to clean the RCNode *cur* from the linked list.
2. If the *next* of *cur* is **null** (line 43), i.e. *cur* is the node  $RTail$ , a new RCNode *RN* is allocated and added. After that, the method *SnapRange* (line 28 to line 33) is called to collect the snapshot of the desired subset of the data-structure.

In an execution of *SnapRange*, the collection of nodes from the data-structure and storing them in the list  $L$

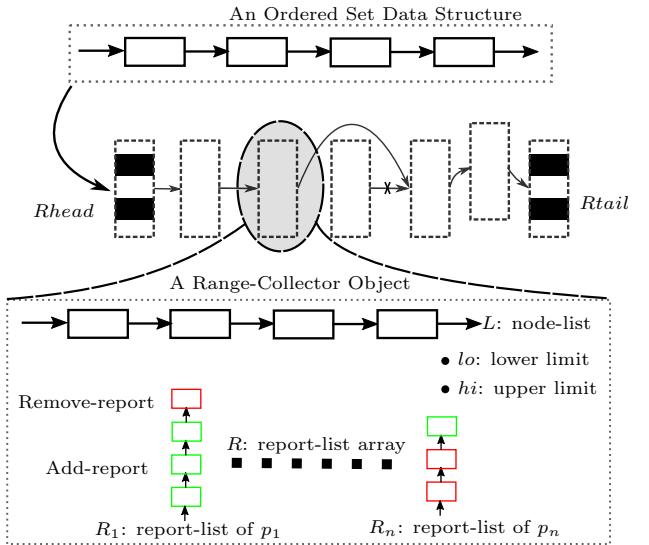


Figure 2: A range-collector implementation

works exactly the same way as in a snap-collector. We begin with finding the node with smallest key in the data structure using the method *FindMinNode*, at line 14. In linear data structures such as linked list and skip list, it is straightforward: the first node (head or the successor of it in case head is a dummy node) is the node with the smallest key. In a BST we need to traverse down to locate the node with smallest key. Similarly, to find the node in the data structure with successor key the method *Next*, line 15, works directly in linear data structure, whereas, in a BST, we need to perform depth first search. The methods *FindMinNode* and *Next* for the lock-free BST are described in the section 2.3.

The method *Deactivate* (line 19 to line 23), which consists of the steps - (a) adding a node with key  $\infty$  to the list  $L$  and (b) a **CAS** to set the mark-bit of *next*, is called to deactivate a range-collector. Setting the mark bit of the *next* works as the linearization point of the *RangeSearch* operation. After this step any call to *IsActive* returns **false**. Clearly, the terminal node of  $L$  with key  $\infty$  causes termination of the collection of any further node. Additionally, the *SnapRange* method also calls the method *BlockFurtherReports* which works along the same lines as in a snap-collector and thus ensures that no further concurrent modify operation can be reported to  $RN$ .

- On completion of *SnapRange*, *mode* is changed to **CLEAN** and the RCNode *RN* is attempted to be detached from the unordered list. If the **CAS** to detach the node fails, the traversal is restarted. If *mode* is set to **CLEAN** then reaching the node  $RTail$  indicates that the targeted RCNode has been detached. Finally, the method *ProcessRange*, line 26, is called to return the desired data-structure-nodes. *ProcessRange* includes steps of sorting the added reports and combining the nodes from  $L$  to produce a set of nodes with keys in the range  $[lo hi]$ .
3. If the *next* of *cur* is neither marked nor **null**, it indicates an active *RangeSearch*, therefore we check the relation between  $([lo hi])$  and  $([x y])$  by calling *ChkOvrlap* and
    - (a) if the relation is *isContained* (line 53), we help the undergoing *SnapRange* at *cur* and *ProcessRange* is used to return only those nodes which have the keys  $\in [lo hi]$ .

- (b) if the relation is `isDisjoint` (line 58), we simply move to the next RSNode as `cur` does not cover any data-structure-node with key in the range  $[lo \ hi]$ .  
(c) and finally, if the relation is `contains` or `overlaps` (line 60), we help the undergoing `SnapRange` at `cur` and the traversal is restarted from `RHead`.

---

**Algorithm 2.** The linearizable range search algorithm

---

**Returns a list of keys in interval  $[lo, hi]$ .**

```

34 RangeSearch(K lo, K hi)
35 | pre=RHead; cur=pre->next; mode=INIT; RN=null;
36 | retry:
37 | while true do
38 |   (ref, m)=cur->next;
39 |   while ref ≠ null and m==1 do
40 |     if !CAS(pre->next, (cur, 0), (ref, 0)) then
41 |       goto retry;
42 |     (ref, m)=cur->next; pre=cur; cur=ref;
43 |   if ref==null then
44 |     if mode==CLEAN then
45 |       return ProcessRange(lo, hi, RN);
46 |     RN=RCNode(true, lo, hi, null, null, (*RTail, 0));
47 |   if CAS(pre->next, (cur, 0), (*RN, 0)) then
48 |     SnapRange(lo, hi, RN);
49 |     (ref, *)=RN->next; mode=CLEAN;
50 |     if CAS(pre->next, (RN, 0), (ref, 0)) then
51 |       return ProcessRange(lo, hi, RN);
52 |     else goto retry;
53 |   else if ChkOverlap(lo, hi, cur)==isContained then
54 |     SnapRange(cur-lo, cur-hi, cur);
55 |     RN=cur; mode=CLEAN; (ref, *)=RN->next;
56 |     if CAS(pre->next, (RN, 0), (ref, 0)) then
57 |       return ProcessRange(lo, hi, RN);
58 |     else if ChkOverlap(lo, hi, cur)==isDisjoint then
59 |       (ref, m)=cur->next; pre=cur; cur=ref;
60 |     else
61 |       SnapRange(cur-lo, cur-hi, cur);
62 |       (ref, *)=cur->next;
63 |       CAS(pre->next, (cur, 0), (ref, 0));

```

---

Most commonly, the lock-free 1-dimensional ordered data structures available in literature [14, 21, 23] have a single successful CAS step as the completion of an Add operation. Similarly, for a Remove operation, a successful CAS is required to inject a “mark” at a connecting link from the node to remove. This CAS step is commonly known as the logical remove step, and after that the logically removed node is eventually cleaned out of the data structure. Further, the Contains operations terminate at a single atomic `read` step after performing the traversal. We have shown a generalized pseudo-code for these operations in algorithm 4. Now we describe how they synchronize with a concurrent `RangeSearch` operation.

---

**Algorithm 3.** The operation `SyncWithRangeQuery`


---

**Finds appropriate active concurrent range-collection to report node x.**

```

64 SyncWithRangeQuery(DSNPtr x, RptType report, int tid)
65 | (ref, *)=RHead->next;
66 | while ref≠RTail do
67 |   if isActive(ref) and ref.lo≤x.key≤ref.lo then
68 |     if report == ADD and IsPresent(x) then
69 |       Report(ref, Report(x, ADD, tid);
70 |     else Report(ref, Report(x, REM, tid));
71 |     break;
72 |   else (ref, *)=ref->next;

```

---

The set operations `Contains` (line 74 to 79), `Add` (line 80 to 93) and `Remove` (line 94 to 104), which are concurrent to a `RangeSearch`, report the data-structure-nodes in a similar

way as they do in [25]. An object **Report** consists of the address of the node to be reported and the type of report (`ADD` or `REM`). To report a node  $x$ , the method `SyncWithRangeQuery`, line 65 to 72, is called. `SyncWithRangeQuery` traverses through the unordered list to locate an `RCNode` which is active and has the range  $\exists key(x)$ . However, during the traversal no `RangeSearch` is helped. Before reporting an `ADD`, the method `IsPresent` is called to check whether  $X$  is logically removed. If no relevant `RCNode` is found then nothing is reported.

A `Contains` (line 74 to line 79) operation on finding the target node reports it as `ADD` if it is not logically removed else `REM` is reported. If the node is not found, there is nothing to report. An `Add` operation (line 80 to line 93) first adds the desired node, then calls the method `SyncWithRangeQuery` to report `ADD`. If a node with query key is found in the data-structure, which is not logically removed, then it behaves as a `Contains` operations. On finding a node with the query key but logically removed, a `REM` is first reported, after that the pending Remove operation is helped and then the `Add` is reattempted. A `Remove` (line 94 to line 104) on finding a node with the query key, attempts to logically remove the node, reports `REM` of the node, then completes the remaining steps to clean the node.

---

**Algorithm 4.** A Lock-free data structure algorithm that employs the linearizable range search algorithm

---

**Returns the DSNNode with key equal key; returns null if no such node present.**

```

73 Find(K key)
    Returns true if there exists a DSNNode with key equal key else false.
74 Contains(K key, int tid)
75 | if (x=Find(key)) == null then return false;
76 | if IsPresent(x) then
77 |   SyncWithRangeQuery(x, ADD, tid); return true;
78 | else
79 |   SyncWithRangeQuery(x, REM, tid); return false;
    Adds DSNNode (key) to return true if no such node present else returns false.
80 Add(K key, int tid)
81 | while true do
82 |   if (x=Find(key))≠null then
83 |     if IsPresent(x) then
84 |       SyncWithRangeQuery(x, REM, tid);
85 |       return false;
86 |     else
87 |       SyncWithRangeQuery(x, REM, tid);
88 |       ...Complete Remove.
89 |       continue;
90 |     else
91 |       ...Complete Add.
92 |       SyncWithRangeQuery(x, ADD, tid);
93 |       return true;
    Removes the DSNNode (key) to return true; returns false if no such node present.
94 Remove(K key, int tid)
95 | if (x=Find(key)) == null then return false;
96 | if IsPresent(x) then
97 |   ...Logically remove DSNNode (key).
98 |   SyncWithRangeQuery(x, REM, tid);
99 |   ...Complete Remove.
100 |   return true;
101 | else
102 |   SyncWithRangeQuery(x, REM, tid);
103 |   ...Complete Remove.
104 |   return false;

```

---

### 2.3 Case of a lock-free binary search tree

The BST that we used in this work is a modified form of the lock-free external BST of Natarajan et al. [23]. We included parent pointer in the node structure to facilitate depth first search (DFS). See line 105 to line 107 in the algorithm 5 and the fig. 3.

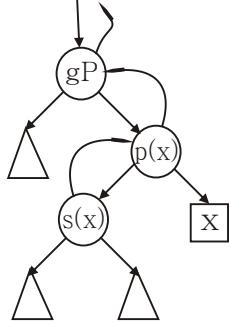


Figure 3: A sub-tree of an external BST with parent pointers

Because of maintaining the parent pointers in nodes, obviously, the Add and Remove operations become complex. The alternate approach to perform DFS is by way of maintaining a local stack in a thread. However, to perform a coordinated scan of the data structure, in which given any node, we find its successor, we can not use the method of stack for DFS. Using a stack will always require a scan to start from the root of the BST and if there is already a concurrent RangeSearch that has almost completed the range scan and stored the nodes in the  $L$  list in an RCNode, we will have large wastage of work. To avoid that, we use parent pointers.

In an instance of **Window**, line 108, we store the address of a leaf-node, which contains data in an external BST, along with the address of its parent. Given these addresses we can always find the successor of the leaf-node. Given the lower limit of a target range, the method **FindMinNode** here, line 109 to 119, returns the **Window** that contains the first node to be stored in the list  $L$  as required in the method **SnapRange** at the line 29. The method **Next** for the BST called at line 32 in **SnapRange** performs one step of the DFS and is given in line 120 to 130.

The **Add**, **Remove** and **Contains** methods in this lock-free BST design are similar to those in the lock-free kd-Tree by Chatterjee et al. [10]. For the description of the **RangeSearch** method, which is the main topic of this paper, we do not require the description of those methods. An interested reader may see those descriptions in [10].

## 3. CORRECTNESS PROOF

We prove that our algorithm is linearizable and lock-free. We assume that a reader is familiar with the basic definitions of execution history, equivalence of histories, linearizability and the related notions. For reference one can refer to [8].

First we list out the linearization points of the ADT operations as presented in the generalized pseudo-code of algorithms 2 and 4. We already mentioned that the linearization point of a **RangeSearch** operation is the atomic **CAS** step at line 22, where the *next* of the RCNode is marked in the method **Deactivate** by one of the processes performing **SnapRange** at it. The introduction of **RangeSearch** in a linearizable partial implementation of the ADT OSet with

---

**Algorithm 5.** DFS implementation in the lock-free binary search tree.

---

**Structure of a BST node.**

```
105 struct Node {
106     K key; NPtr lt, rt, parent;
107 } *NPtr;
```

**Packet of address of a leaf-node and its parent.**

```
108 struct Window {NPtr cur; NPtr par;};
```

**Returns the Window containing the node with key just  $\geq$  lo in the BST; returns null if no such node present.**

```
109 FindMinNode(K lo)
110     prev = root, cur = root->lt;
111     while true do
112         while (cur->lt != null) do
113             | prev = cur; turn = (key < prev->key);
114             | cur = turn ? prev->lt : prev->rt;
115             | if cur->key < lo and cur->key < prev->key then
116                 | | cur = prev->rt;
117             | if (cur->lt == null) then break;
118             | if cur->key < lo then return null;
119             | else return Window(cur, prev);
```

**Returns a Window of with BST node with key just  $\geq$  key( $x \rightarrow cur$ ).**

```
120 Next(Window x)
121     prev = x.par, cur = x.cur;
122     if cur->key < prev->key then cur = prev->rt;
123     else
124         | gP = prev->parent;
125         | while pre->key < gP->key do
126             | | prev = gP; gP = prev->parent;
127             | | prev = gP; cur = prev->rt;
128         | while (cur->lt != null) do
129             | | prev = cur; cur = prev->lt;
130     return Window(cur, prev);
```

Add, Remove and Contains operations does not alter the usual linearization points of these operations, if the output of **RangeSearch** includes a modification performed or observed by such a concurrent operation. The linearization points for these operations in the generalized case are as below.

For a successful **Add** operation, execution of the **CAS**, where a new data structure node is added, is the linearization point. For an unsuccessful **Add** and a successful **Contains**, it is at the point where we read the address of the node with matching key in the method **Find**. For a successful **Remove** operation, the **CAS** of the logical remove step is the linearization point. Linearization arguments for an unsuccessful **Remove** and a similar **Contains** have two cases - (a) if there existed a node containing the query key in the data structure at the invocation but was logically or completely removed by a concurrent **Remove** operation before the return of **Find**, the linearization point is placed just after the linearization point of that **Remove** operation (b) if no node containing the query key existed in the data structure at the invocation of the **Remove** or **Contains**, the invocation point itself is taken as the linearization point.

However, if the output of the **RangeSearch** does not include a modification either performed or observed by a concurrent such operation, we linearize the concurrent operation just after the **RangeSearch**. In effect, the linearization points of all the concurrent operations are *anchored* at the linearization point of the **RangeSearch**, which is the atomic **CAS** step at line 22, if the output of the **RangeSearch** misses the observed or performed modification by a concurrent **Add**, **Remove** or **Contains** operations. We can order the linearization points anchored at the same **CAS** step in any arbitrary order, for example, in the order of their invocation points.

Now, because our range search algorithm is independent of

the operations **Add**, **Remove** and **Contains**, we shall prove the linearizability of the full implementation  $\mathcal{I}_M$ , where  $M = \{\text{Add}, \text{Remove}, \text{Contains}, \text{RangeSearch}\}$ , of the ADT OSet by building upon a linearizable partial implementation  $\mathcal{I}_O$ , where  $O = \{\text{Add}, \text{Remove}, \text{Contains}\}$ .

**Theorem 1.** (*Correctness*) *The operations Add, Remove, Contains and RangeSearch are linearizable with respect to the ADT OSet.*

*Proof.* Let  $\alpha$  be an arbitrary execution of the implementation  $\mathcal{I}_M$ . Let  $\mathcal{H}$  be an arbitrary history of  $\alpha$ . We show that a sequential history  $\mathcal{S}$  obtained by following the steps: (a) in  $\mathcal{H}$  append appropriate response (in any arbitrary order) of all the operations which have performed their linearization steps as stated above to obtain  $\text{ext}(\mathcal{H})$ , (b) drop the invocation steps without a matching response to obtain  $\text{complete}(\text{ext}(\mathcal{H}))$ , and (c) construct  $\mathcal{S}$  by arranging the invocation-response pair of operations according to their linearization points, is consistent.

Let  $\rho = \mathcal{H}|_O$  and  $\sigma = \mathcal{S}|_O$  be the projections of  $\mathcal{H}$  and  $\mathcal{S}$  such that they contain operations belonging to only  $O$ . Please note that, for any operation  $op \in \sigma$ ,  $op(k)$ , where  $k \in \mathbb{R}$ , returns either **true** or **false**, whereas, for a **RangeSearch** operation,  $\text{RangeSearch}(x, x')$ , where  $x, x' \in \mathbb{R}$ , is a countably finite subset of  $\mathbb{R}$ .

Now, by the assumption that  $\mathcal{I}_O$  is linearizable, the sequential history  $\sigma$  is consistent. That directly implies that in  $\sigma$  and for a key  $k \in \mathbb{R}$ ,

- (a) if  $op_1, op_2 \in \sigma$  and  $op_1(k) \rightarrow_{\sigma} op_2(k)$  and  $op_1 = \text{Add}$  and  $op_2 = \text{Add}$  and  $op_1(k) = op_2(k) = \text{true}$  then  $\exists op_3 \in \sigma$  s.t.  $op_1(k) \rightarrow_{\sigma} op_3(k) \rightarrow_{\sigma} op_2(k)$  and  $op_3 = \text{Remove}$  and  $op_3(k) = \text{true}$ .
- (b) if  $op_1 \in \sigma$  and  $op_1 = \text{Add}$  and  $op_1(k) = \text{false}$  then  $\exists op_2 \in \sigma$  and  $op_2 = \text{Add}$  and  $op_2(k) = \text{true}$  and  $\nexists op_3 \in \sigma$  and  $op_3 = \text{Remove}$  and  $op_3(k) = \text{true}$  s.t.  $op_1(k) \rightarrow_{\sigma} op_3(k) \rightarrow_{\sigma} op_2(k)$ .
- (c) if  $op_1, op_2 \in \sigma$  and  $op_1(k) \rightarrow_{\sigma} op_2(k)$  and  $op_1 = \text{Remove}$  and  $op_2 = \text{Remove}$  and  $op_1(k) = op_2(k) = \text{true}$  then  $\exists op_3 \in \sigma$  s.t.  $op_1(k) \rightarrow_{\sigma} op_3(k) \rightarrow_{\sigma} op_2(k)$  and  $op_3 = \text{Add}$  and  $op_3(k) = \text{true}$ .
- (d) if  $op_1, op_2 \in \sigma$  and  $op_1(k) \rightarrow_{\sigma} op_2(k)$  and  $op_1 = \text{Remove}$  and  $op_2 = \text{Remove}$  and  $op_1(k) = \text{false}$  and  $op_2(k) = \text{false}$  then either (a)  $\nexists op_3 \in \sigma$  s.t.  $op_1(k) \rightarrow_{\sigma} op_3(k) \rightarrow_{\sigma} op_2(k)$  and  $op_3 = \text{Add}$  and  $op_3(k) = \text{true}$  or (b)  $\nexists op_3 \in \sigma$  s.t.  $op_3(k) \rightarrow_{\sigma} op_1(k)$  and  $op_3 = \text{Add}$  and  $op_3(k) = \text{true}$ .
- (e) if  $op_1 \in \sigma$  and  $op_1 = \text{Contains}$  and  $op_1(k) = \text{true}$  then  $\exists op_2 \in \sigma$  and  $op_2 = \text{Add}$  and  $op_2(k) = \text{true}$  and  $\nexists op_3 \in \sigma$  and  $op_3 = \text{Remove}$  and  $op_3(k) = \text{true}$  s.t.  $op_2(k) \rightarrow_{\sigma} op_3(k) \rightarrow_{\sigma} op_1(k)$ .
- (f) if  $op_1, op_2 \in \sigma$  and  $op_1(k) \rightarrow_{\sigma} op_2(k)$  and  $op_1 = \text{Remove}$  and  $op_2 = \text{Contains}$  and  $op_1(k) = \text{true}$  and  $op_2(k) = \text{false}$  then  $\exists op_3 \in \sigma$  s.t.  $op_1(k) \rightarrow_{\sigma} op_3(k) \rightarrow_{\sigma} op_2(k)$  and  $op_3 = \text{Add}$  and  $op_3(k) = \text{true}$ .

The above expressions describe the consistency of the sequential history  $\sigma$  in the sense that in between two successful **Add** operations with same key there must be a successful **Remove** of that key, or, for two successful **Remove** with same key there must be a successful **Add** of that key or for a successful **Contains** operation with a given key there must be a prior successful **Add** with the same key, and so on. Here by success we mean a **Contains/ Add/ Remove** operation returning **true** in its response.

Now, with these assumptions in place, we need to show that  $\rho$  is consistent. Let  $\rho_n$  be a sub-history of  $\rho$  that contains the first  $n$  complete operations. Let  $\mathbb{A}_n$  be the dataset

which was added to the data structure by the successful **Add** operations in  $\rho_n$ . Let  $\mathbb{B}_n$  be the dataset which was removed from the data structure by the successful **Remove** operations in  $\rho_n$ . Let  $\mathbb{C}_n = \mathbb{A}_n / \mathbb{B}_n$ . We use (strong) induction on  $n$  to show that  $\rho_n$  is consistent  $\forall n \geq 1$ .

Suppose that  $\rho_n$  is consistent  $\forall n : 1 \leq n \leq i$ . Let the  $(i+1)^{\text{th}}$  operation in  $\rho_n$  be  $op$ , where  $x, x' \in \mathbb{R}$ . Then for  $\rho_{i+1}$  we prove the following:

1. Let  $op$  be  $\text{RangeSearch}(x, x')$  and  $\text{RangeSearch}(x, x') \ni k$ .

- (a) We show that if  $\exists op_1 \in \rho_i$  and  $op_1(k) \rightarrow_{\sigma} op(k)$  and  $op_1 = \text{Add}$  and  $op_1(k) = \text{true}$  then  $\nexists op_2 \in \rho_i$  and  $op_2 = \text{Remove}$  and  $op_2(k) = \text{true}$  s.t.  $op_1(k) \rightarrow_{\sigma} op_2(k) \rightarrow_{\sigma} op(k)$ .

*Proof.* Suppose there  $\exists$  such an  $op_2 \in \rho_i$ . Then by the construction of the linearization points, the  $\text{RangeSearch}(x, x')$  must have noticed this **Remove** operation. That implies that either the **Remove** operation itself would have reported the logical removal of the node containing  $k$  or the **RangeSearch** operation started after **Remove** and they were not concurrent. That implies that  $\text{RangeSearch}(x, x') \not\ni k$ , which is a contradiction.

- (b) We show that if  $\exists op_1 \in \rho_i$  and  $op_1(k) \rightarrow_{\sigma} op(k)$  and  $op_1 = \text{Contains}$  and  $op_1(k) = \text{true}$  then  $\nexists op_2 \in \rho_i$  and  $op_2 = \text{Remove}$  and  $op_2(k) = \text{true}$  s.t.  $op_1(k) \rightarrow_{\sigma} op_2(k) \rightarrow_{\sigma} op(k)$ .

*Proof.* Same as (a) above.

- (c) We show that if  $\exists op_1 \in \rho_i$  and  $op_1(k) \rightarrow_{\sigma} op(k)$  and  $op_1 = \text{Add}$  and  $op_1(k) = \text{true}$  then  $\nexists op_2 \in \rho_i$  and  $op_2 = \text{Contains}$  and  $op_2(k) = \text{false}$  s.t.  $op_1(k) \rightarrow_{\sigma} op_2(k) \rightarrow_{\sigma} op(k)$ .

*Proof.* Suppose there  $\exists$  such an  $op_2 \in \rho_i$ . Then by the algorithm either the **Contains** operation found a node with key  $k$  as logically removed or it did not find it at all. If it found the node logically removed then by the construction of the linearization points, the  $\text{RangeSearch}(x, x')$  must have either been notified of the same or the **RangeSearch** started after the completion of **Contains** and they were not concurrent. That implies that  $\text{RangeSearch}(x, x') \not\ni k$ , which is a contradiction. On the other hand, if **Contains** did not find this node then there must have been a successful **Remove** operation after the **Add** and before the **Contains**. However, that is not possible by (a) above, hence contradiction.

2. Let  $op \in \mathcal{O}$  then by the consistency of  $\sigma$  mentioned before, they are consistent among themselves.

(1) and (2) together prove that  $\rho_{i+1}$  is consistent. Thus,  $\rho_n$  is consistent for  $n = i+1$  provided  $\rho_i$  is consistent. Hence, by (strong) induction  $\rho_n$  is consistent for all  $n \geq 1$ .  $\square$

Proving lock-freedom of the full implementation  $\mathcal{I}_M$  of OSet is straight-forward. Because  $\mathcal{I}_O$  is lock-free, at least one thread finishes its operation if none of them are performing **RangeSearch**. Now, even if **RangeSearch** is performed, the operations in the set  $\mathcal{O}$  perform a single **CAS** for reporting without any failure-retry and thus even if one thread finishes its operation in  $\mathcal{I}_O$ , it must finish its operation in  $\mathcal{I}_M$  as well. With regard to concurrent **RangeSearch** operations in  $\mathcal{I}_M$ , they do not perform any write in the data structure. Further, in the augmented list, whenever a **CAS** fails to add a new **RCNode**, the traversal during the reattempt starts from the **RHead** and any pending operation is helped to finish. Thus, for concurrent **RangeSearch** operations, it can not be possible that no operation finishes in finite number of steps.

Thus we have the following theorem.

**Theorem 2.** (*Liveness*) *The operations Add, Remove, Contains and RangeSearch are lock-free.*

This concludes the proof of the presented algorithm.

## 4. EXPERIMENTAL EVALUATION

We implemented the presented algorithms in Java using RTTI. We evaluated the introduced range search algorithm in three lock-free data-structures - (a) H\_LinkedList: linked list of [14], (b) Im\_Ex\_BST: a lock-free external BST in which a non-recursive traversal is facilitated using parent-links as described in section 2.3 (c) ConcSkipList: the skip list of [21].

We thankfully obtained the basic code of [25] from the authors in a personal communication. They used **AtomicMarkedReference** objects of `java.util.concurrent.atomic` library in the snap-collector implementation as well as in the linked list of [14], whereas the java library code of [21] was used for the skip list. To optimize the code, we aligned all the implementations to RTTI by replacing every instance of **AtomicMarkedReference** object with a volatile variable and used **AtomicReferenceFieldUpdater** on top of it for **CAS**.

We compared the algorithm with the range search implementation of [6] (BA\_KST64), which is based on Java RTTI. We used the author's code available at their home-page. In the experiments in [6],  $k = 64$  produced the best results among the k-ary search trees. Therefore, we chose to compare our implementation with the one with  $k = 64$ .

To simulate the variation due to the contention, we selected the combination of key-range, percentage of operations and the number of threads as following: (a) the % of (Add, Remove, Contains, RangeSearch)  $\in \{(05, 05, 89, 01), (05, 05, 88, 02), (25, 25, 89, 01), (25, 25, 88, 02)\}$ ; (b)  $|\{key \in K\} \in \{10^2, 10^3\}$  and (c) the number of threads  $\in \{2, 4, 8, 16, 28, 32\}$ .

We used a machine with a dual chip Intel(R) Xeon(R) E5-2695 v3 processor with 14 hardware threads per chip (28 hardware threads in total with hyper-threading) running at 2.30 GHz. The machine has 64 GB of RAM and runs over CentOS Linux 7.1.1503 (Kernel version: 3.10.0-229.el7.x86\_64) with Java HotSpot(TM) 64-Bit Server VM (1.8.0\_51) with 1 GB initial heap size and 15.6 GB maximum heap size. All the implementations were compiled using `javac` version 1.8.0\_51 and the runtime flags `-d64 -server` were used. We performed 10 repetitions of 5 seconds runs for each combination of the parameters shown in the graphs. The average over the 10 trials are recorded. The keys in all the data-structures are taken of Integer type. We ran the experiments for up to 32 threads to observe the scalability above the thread saturation limit - 28 threads, of the processor.

In the experiments, the keys are selected at random from the chosen key-ranges following a uniform distribution. Additionally, all the threads in the experiment perform all the operations selecting next operation at random with a probability expressed by the distribution percentage. In RangeSearch experiments, we recorded the throughput of the method `Size` which computes the size of the node-set returned by a call of `RangeSearch`. To call a `RangeSearch` (`[lo, hi]`), we randomly selected two keys from the chosen key-range and passed their min as `lo` and their max as `hi`.

For a linearizable `Size` method in BA\_KST64, we used the length of the return of the `subset()` method thereof.

## 4.1 Observations and Discussion

Figure 4 demonstrates the experimental observations. We observed a completely different behavior of our implementation compared to that of BA\_KST64 with regards the performance and overhead.

- With the growth in the number of modify operations (compare the plots of columns 1 and 2 together to 3 and 4), our method substantially outperforms BA\_KST64 (even linked list performs better than KST64 in smaller data-structure with high modification percentage). This is expected from the growing number of validations required in BA\_KST64 with the growth in modify operations.
- In all the cases, as the number of threads increases, our method exhibits good scalability whereas it is opposite in BA\_KST64. This again can be understood in terms of increasing number of validations required in high contention scenarios in BA\_KST64.
- Among the data-structures considered for evaluating our generic method, in high contention cases BST outperforms the skip list, whereas in cases of low contention and smaller size of the data structure, skip list wins over the BST. This can be explained in terms of higher number of steps required to find the successor of a node in an external BST.
- On increasing the percentage of RangeSearch operations (compare the plots of column 1 with 2 and 3 with 4), the throughput of our algorithm decreases across the data-structure types, whereas we do not see similar throughput change in BA\_KST64. It indicates that our method has marginally higher overhead compared to BA\_KST64, specifically when the number of concurrent threads and percentage of modify operations is low. It can be explained in terms of the fundamental difference in the snapshot collection strategies ([1] vs. [20]). We make Contains operations report to the concurrent RangeSearch, whereas in BA\_KST64, Contains do not bother about RangeSearch. The experimental evaluations in [25] also showed somewhat similar behaviour with respect to the comparison between throughput and overhead of [25] and [26].

## 5. CONCLUSION

In this paper we presented a generic method to perform linearizable range search in lock-free 1-dimensional ordered data structures. Our experiments showed that the proposed range search method scales well in high contention scenarios.

The k-ary tree by Brown et al. [6] used dirty bits in nodes. We observed that this method achieves better throughput in low contention scenarios compared to our method. We can design a hybrid method that uses similar strategy to achieve scalability of RangeSearch together with lower overhead in low contention scenarios, whereas falls back to the presented method when percentage of concurrent modification is high. Also, we can explore the design of a lock-free BST which facilitates faster computation of successor of a given node, for instance, by connecting the leaves.

## 6. REFERENCES

- [1] Y. Afek, H. Attiya, D. Dolev, E. Gafni, M. Merritt, and N. Shavit. Atomic snapshots of shared memory. *Journal of the ACM (JACM)*, 40(4):873–890, 1993.

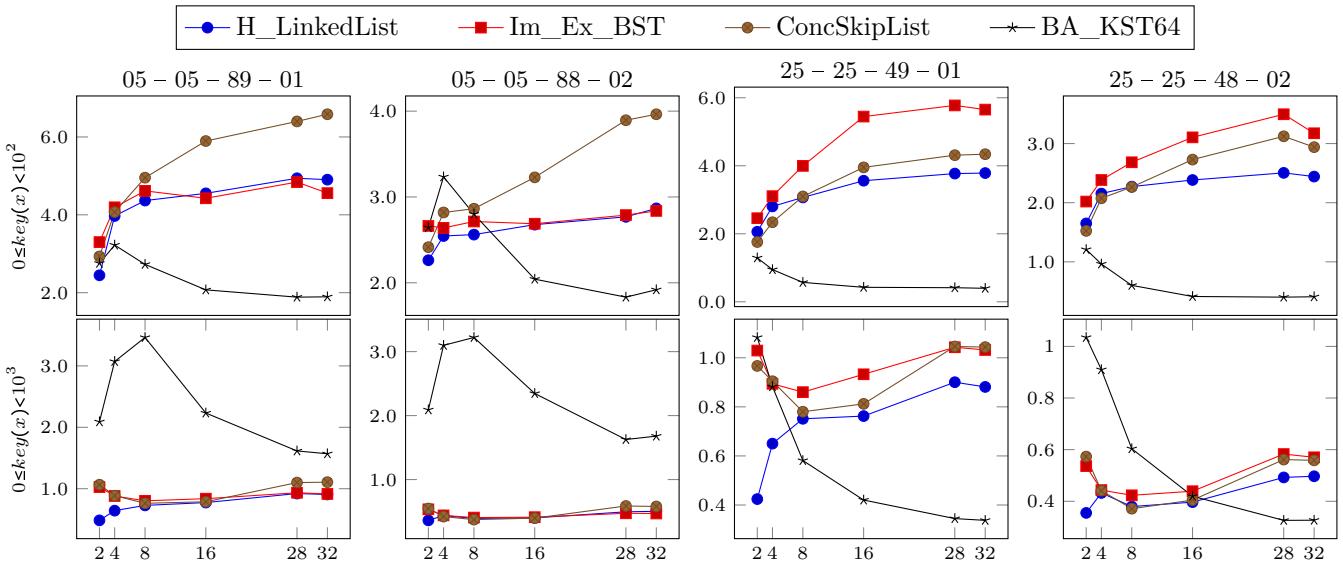


Figure 4: Performance in terms of the throughput ( $\frac{1000 \times \#Ops}{ms}$ ) vs.  $\#threads$ , varying the range of the keys and the distribution of operations in terms of Add %–Remove %–Contains %–RangeSearch %.

- [2] Y. Afek, G. Stupp, and D. Touitou. Long-lived adaptive collect with applications. In *40th FOCS*, pages 262–272, 1999.
- [3] H. Attiya, R. Guerraoui, and E. Ruppert. Partial snapshot objects. In *20th SPAA*, pages 336–343, 2008.
- [4] H. Avni, N. Shavit, and A. Suissa. Leaplist: lessons learned in designing tm-supported range queries. In *32nd PODC*, pages 299–308. ACM, 2013.
- [5] D. Basin, E. Bortnikov, A. Braginsky, G. Golan Gueta, E. Hillel, I. Keidar, and M. Sulamy. Brief announcement: A key-value map for massive real-time analytics. In *35th PODC*, pages 487–489, 2016.
- [6] T. Brown and H. Avni. Range queries in non-blocking k-ary search trees. In *16th OPODIS*, pages 31–45. Springer, 2012.
- [7] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):4, 2008.
- [8] B. Chatterjee. Lock-free 1-dimensional range queries. Technical Report 2015:06, ISNN 1652-926X, Chalmers University of Technology, 2015.
- [9] B. Chatterjee, N. Nguyen, and P. Tsigas. Efficient lock-free binary search trees. In *33rd PODC*, pages 322–331, 2014.
- [10] B. Chatterjee, I. Walulya, and P. Tsigas. Concurrent linearizable nearest neighbour search in lockfree-kd-tree. Technical report, Chalmers University of Technology, 2015.
- [11] F. Ellen, P. Fatouros, J. Helga, and E. Ruppert. The amortized complexity of non-blocking binary search trees. In *33rd PODC*, pages 332–341, 2014.
- [12] F. Ellen, P. Fatouros, E. Ruppert, and F. van Breugel. Non-blocking binary search trees. In *29th PODC*, pages 131–140, 2010.
- [13] M. Fomitchev and E. Ruppert. Lock-free linked lists and skip lists. In *23rd PODC*, pages 50–59, 2004.
- [14] T. L. Harris. A pragmatic implementation of non-blocking linked-lists. In *15th DISC*, pages 300–314, 2001.
- [15] A. HBase. A distributed database for large datasets. *The Apache Software Foundation, Los Angeles, CA*. URL <http://hbase.apache.org>, 4(4.2).
- [16] M. Herlihy. Wait-free synchronization. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 13(1):124–149, 1991.
- [17] M. P. Herlihy and J. M. Wing. Linearizability: A

- correctness condition for concurrent objects. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 12(3):463–492, 1990.
- [18] S. V. Howley and J. Jones. A non-blocking internal binary search tree. In *24th SPAA*, pages 161–171, 2012.
- [19] D. Imbs and M. Raynal. Help when needed, but no more: efficient read/write partial snapshot. *Journal of Parallel and Distributed Computing*, 72(1):1–12, 2012.
- [20] P. Jayanti. An optimal multi-writer snapshot algorithm. In *37th STOC*, pages 723–732, 2005.
- [21] D. Lea. *ConcurrentSkipListMap*. In *java.util.concurrent*.
- [22] M. M. Michael and M. L. Scott. Simple, fast, and practical non-blocking and blocking concurrent queue algorithms. In *15th PODC*, pages 267–275, 1996.
- [23] A. Natarajan and N. Mittal. Fast concurrent lock-free binary search trees. In *19th PPoPP*, pages 317–328, 2014.
- [24] N. Nguyen, P. Tsigas, and H. Sundell. Parmarksplit: A parallel mark-split garbage collector based on a lock-free skip-list. In *International Conference on Principles of Distributed Systems*, pages 372–387. Springer, 2014.
- [25] E. Petrank and S. Timnat. Lock-free data-structure iterators. In *Distributed Computing*, pages 224–238. Springer, 2013.
- [26] A. Prokopec, N. G. Bronson, P. Bagwell, and M. Odersky. Concurrent tries with efficient non-blocking snapshots. In *Acm Sigplan Notices*, volume 47, pages 151–160. ACM, 2012.
- [27] K. F. Sagonas and K. Winblad. Efficient support for range queries and range updates using contention adapting search trees. In *Languages and Compilers for Parallel Computing - 28th International Workshop, LCPC 2015, Raleigh, NC, USA, September 9-11, 2015, Revised Selected Papers*, pages 37–53, 2015.
- [28] H. Samet. *Foundations of multidimensional and metric data structures*. Morgan Kaufmann, 2006.
- [29] H. Sundell and P. Tsigas. Fast and lock-free concurrent priority queues for multi-thread systems. *J. Parallel Distrib. Comput.*, 65(5):609–627, 2005.
- [30] H. Sundell and P. Tsigas. Lock-free and practical doubly linked list-based deques using single-word compare-and-swap. In *9th OPODIS*, pages 240–255. Springer, 2005.

# Efficient Algorithms for Predicate Detection using Hybrid Logical Clocks

Sorrachai  
Yingcharonthawornchai<sup>1</sup>  
yingchar@cse.msu.edu

Eric Torng<sup>1</sup>  
torng@cse.msu.edu

Vidhya Tekken Valapil<sup>1</sup>  
tekkenva@cse.msu.edu

Murat Demirbas<sup>2</sup>  
demirbas@buffalo.edu

Sandeep Kulkarni<sup>1</sup>  
sandeep@cse.msu.edu

<sup>1</sup>Department of Computer Science and Engineering  
Michigan State University  
East Lansing MI 48824  
<sup>2</sup>Department of Computer Science and Engineering  
University at Buffalo, The State University of New York  
Buffalo NY 14260-2500

## ABSTRACT

Predicate detection is a critical aspect in debugging and monitoring of distributed systems. Runtime monitoring of distributed systems should satisfy two main requirements: wait-free execution and efficient computation. Monitoring is wait-free if it is non-intrusive (application being monitored is not stopped due to monitoring) and is efficient if the monitoring overhead is small. We define a monitor to be  $\epsilon$ -sensitive if it is guaranteed to detect all predicates that are true for at least an  $\epsilon$  time interval in true physical time. We focus on using hybrid logical clocks (HLCs) to perform wait-free and efficient predicate detection. HLC uses only constant space, has no false positives, and is  $2\epsilon$ -sensitive where  $\epsilon$  is the synchronization error bound. We focus on developing efficient algorithms for detecting weak conjunctive predicates (WCPs) with the help of HLC and then extend them to detect arithmetic predicates such as those necessary for expressing resource usage, network density, and so on.

## 1. INTRODUCTION

### 1.1 Motivation

Monitoring a distributed program at runtime to detect whether it satisfies/violates a specific property is an important primitive for building highly scalable, highly available, and highly auditable large-scale web services. Practical best effort monitoring applications have been developed by Facebook, Twitter and Google to monitor their own services [3, 15, 16]. Formal monitoring systems with provable performance guarantees have been studied by the distributed systems community under the “distributed pred-

icate detection problem” title for many decades [1, 7, 8]. Detecting a global predicate in a distributed system is challenging because without perfect synchronization, there are many possible sequences of global states. This is especially challenging if the global predicate is only satisfied for a very small interval of true physical time. Intuitively, global predicates that are true for longer intervals of true physical time should be easier to detect. *We define a monitor to be  $\epsilon$ -sensitive if it is guaranteed to detect any global predicate that is true for at least an  $\epsilon$  interval of true physical time.* We argue that  $\epsilon$ -sensitivity makes sense given the widespread adoption of synchronization protocols such as NTP [13] that ensure each local process clock is within  $\epsilon$  of the true physical time.

Detecting a global predicate in a distributed system is challenging because without perfect synchronization, there are many possible sequences of global states. This is especially challenging if the global predicate is only satisfied for a very small interval of true physical time. Intuitively, global predicates that are true for longer intervals of true physical time should be easier to detect. *We define a monitor to be  $\epsilon$ -sensitive if it is guaranteed to detect any global predicate that is true for at least an  $\epsilon$  interval of true physical time.* We argue that  $\epsilon$ -sensitivity makes sense given the widespread adoption of synchronization protocols such as NTP [13] that ensure each local process clock is within  $\epsilon$  of the true physical time.

Detecting a global predicate in a distributed system is challenging because without perfect synchronization, there are many possible sequences of global states. This is especially challenging if the global predicate is only satisfied for a very small interval of true physical time. Intuitively, global predicates that are true for longer intervals of true physical time should be easier to detect. *We define a monitor to be  $\epsilon$ -sensitive if it is guaranteed to detect any global predicate that is true for at least an  $\epsilon$  interval of true physical time.* We argue that  $\epsilon$ -sensitivity makes sense given the widespread adoption of synchronization protocols such as NTP [13] that ensure each local process clock is within  $\epsilon$  of the true physical time.

Our goal is to provide non-intrusive and efficient  $\epsilon$ -sensitive runtime monitoring of distributed systems. A non-intrusive mechanism should provide wait-free execution which means that the act of monitoring never blocks a distributed program. An efficient mechanism should monitor predicate satisfaction using both small timestamps and minimal resources. Developing runtime monitoring mechanisms that are both non-intrusive and efficient is very difficult.

### 1.2 Related Work

Most existing work on distributed system monitoring has focused on asynchronous systems that use vector clocks (VCs) [6, 12, 14]. Asynchronous monitors are appealing because they can be non-intrusive; in particular, they make minimal assumptions about the underlying system and network. Unfortunately, asynchronous monitors have two sources of inefficiency that limit their scalability and impede their adoption in real systems. First, general predicate detection with asynchronous monitors is NP-complete [2], although polynomial time algorithms exist for special cases of predicate detection such as linear predicates and conjunctive predicates [2, 9]. Second, asynchronous monitors require  $\Theta(n)$  space VC timestamps to track all causalities in the system where  $n$  is the number of processes. Thus, each message has linear size in the system which limits scalability of asynchronous monitors.

One way to avoid the overhead of VCs is to use physical

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ICDCN '17, January 04-07, 2017, Hyderabad, India

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007780>

time which has  $O(1)$  size timestamps. For example, Spanner [4] uses TrueTime that is based on highly synchronized physical clocks. However, it requires delaying certain events to ensure that the timestamp of event  $f$  that causally depends on event  $e$  is higher than that of event  $e$ . In other words, the pure physical time model (PT) is not wait-free; it introduces delays in the programs being monitored.

Between these two extremes lie the recently introduced hybrid time models that incorporate elements of both logical and physical time models [10,17]. Specifically, Hybrid logical clocks (HLCs) and Hybrid vector clocks (HVCs) generalize logical clocks (LC) and vector clocks (VC), respectively, by introducing a physical time synchronization bound  $\epsilon$  where we assume that the loosely-synchronized clock protocols like NTP allow us to assume that HLC/HVC timestamps are within some interval  $\epsilon$  of actual physical time. HLC generalizes LC and, similar to LC, captures one-way causality and has  $O(1)$  size timestamps. HVC generalizes VC and, similar to VC, captures two-way causality. While the worst case size of HVC timestamps is  $O(n)$ , in many systems, a smaller clock size suffices. Similar to LC, HLC-based detection may suffer from false negatives in that it may fail to detect instances where the global predicate is satisfied whereas HVC-based detection may suffer from false positives in that it may report instances where the global predicate is not true [18]. However, a simple observation is that if the synchronization error is  $\epsilon$ , then HLC is  $2\epsilon$ -sensitive; that is, HLC is guaranteed to detect any predicate that is true for at least a  $2\epsilon$  interval in true physical time.

Since HLC based detection has low overhead ( $O(1)$  size timestamps), we focus on using HLC in predicate detection. HLCs overcome some of the drawbacks of PTs and VCs. Unlike PTs, HLC is very resilient to loss/lapse of clock synchronization. Specifically, if the synchronization bound  $\epsilon$  increases for some amount of time, the sensitivity of HLC will potentially decrease until synchronization is reestablished. In essence, the sensitivity of HLC-based monitoring automatically adjusts to the accuracy of the synchronization. HLCs also preserve the advantages that PTs have over VCs. HLCs are more efficient than VCs requiring only  $O(1)$  scalar space as compared to  $O(n)$  space and potentially less time to perform predicate detection. HLCs are always close to physical clocks which enables them to be used in place of physical time; for example, HLC can be used to query the system in relation with physical time.

Although HLC provides the potential to provide efficient detection, there is no known efficient algorithm for solving predicate detection with HLC for a wide variety of different environments (offline/online, different types of predicates and so on). *Our goal is to fill this gap by providing efficient predicate detection algorithms and data structures for as wide a range of predicates and environments as possible given HLC.*

### 1.3 Proposed Approach

Our key insight for achieving efficient detection with HLCs as well as for reducing the false-negative rate of HLCs, is to employ interval predicates. Each local process can compute intervals where the predicate is true for that process. This implies that we can solve the distributed predicate detection problem by having a monitor collect all intervals from the local processes and searching these intervals to determine if there exists an HLC timestamp that is included in at least one reported interval from every process, and we

can solve other predicate detection problems using similar interval processing algorithms. All of our algorithms are  $2\epsilon$ -sensitive where  $\epsilon$  is the synchronization error bound.

With this key insight, we present a general framework for efficiently solving a variety of predicate detection problems with HLC. We begin with detection of WCPs (Weak conjunctive predicates). WCP is an important type of predicate in distributed system since it allows one to express important constraints (e.g., lack of a leader) in distributed system. In addition to WCPs, our framework can also handle arithmetic predicates that are of the form  $f(x_1, \dots, x_n) \leq C$ , where  $C$  is a constant,  $x_j$  is a variable at process  $j$ , and  $f$  is an arbitrary arithmetic function. For example, an ISP may limit total download speed from nodes in the cloud from a malicious user. We can set  $f(x_1, \dots, x_n)$  to be the sum of all downloads in each node from a user where  $x_i, 1 \leq i \leq n$  represents current download speed of the user.

Our framework can also be extended to model reasonable assumptions about message delivery and message delays.

### 1.4 Contributions and Organization of the Paper

Based on our framework, we present several algorithms for detecting WCPs and arithmetic predicates. These algorithms focus on a system with  $n$  processes each of which reports suitable information to a monitor to determine whether the given predicate is satisfied. For example, for WCP, it reports the intervals where the local predicate is true. Let  $I$  denote the number of intervals reported to the monitor. The properties of our algorithms are as follows (a summary of these results is provided in Table 1). In Section 4, we present a naïve algorithm with worst case complexity  $O(In)$  for detection of WCPs with HLC in an *offline* setting, i.e., where the intervals are available in advance, and the intervals of each process are in a sorted list. In Section 5, we present an improved algorithm that enables *online* detection (intervals may be reported in any order). In Section 6, we show that the online complexity can be further reduced to  $O(I \log n)$  provided intervals reported by each process are guaranteed to arrive at the monitor in FIFO order (arbitrary interleaving among intervals reported by different processes). In Section 7, we show how our algorithms can be used to evaluate arithmetic predicates such as those necessary for expressing resource usage, network density, etc.

## 2. PRELIMINARIES

In this section, we formally define our system model and summarize the key properties of *hybrid logical clocks (HLC)* [10]. We assume the system consists of a set of  $n$  processes where each process has its own clock that is guaranteed to be within  $\epsilon$  of the *ideal global clock* using protocols such as NTP [13]. However, the processes are unaware of the precise/true global clock at any given time. The processes communicate via messages. We make no assumptions about communication between processes. The processes typically communicate with a monitor, and we typically do not make any assumptions about message order or delays to the monitor, but we do show how we can improve the efficiency of our algorithms when we do add message delay or message order assumptions when communicating with the monitor. We clearly identify these assumptions in the respective sections.

HLC combines physical clocks and logical clocks to lever-

Table 1: Summary of Conjunctive Predicate Detection Algorithms with HLC

Section #	Algorithm	Running Time	Assumption
4	Naive	$O(I_n)$	Offline
5	$\mathcal{C}$ -point Tree	$O(I \log I)$	-
6.1	Naive with Minheap	$O(I \log n)$	Local Ordering Property
6.2	Naive with $\Delta$ -Bounded Message Delay	$O(I \log(\Delta n))$	Message Delivery Property

age key benefits of both clocks. Similar to physical clocks, HLC provides a scalar sized logical timestamp that is close to the corresponding physical clock. Similar to logical clocks [11], a message includes the logical timestamp corresponding to its send event, and logical timestamps can be used to infer causality between different events. An HLC timestamp of event  $a$  consists of two parts,  $l.a$  and  $c.a$ , where  $l.a$  represents our best approximation of the global time that  $a$  occurs at. More precisely,  $l.a$  tracks the maximum time event (at any process) that event  $a$  was aware of and  $c.a$  is a bounded counter that encodes causality information. In [10], it has been shown that the theoretical maximum value of  $c$  is  $O(n)$ . In practice, this value remains very small. We set counter  $c.a$  to be larger than counter  $c.b$  to indicate that event  $a$  occurred later than an event  $b$  even though  $l.a = l.b$ . For completeness, we include below the details of how these timestamps are updated when a process either sends/ receives a message to/from another process or creates a local event.

---

**Algorithm 1** HLC Algorithm from [10]

---

**Send/Local Event**

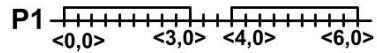
- 1:  $l'.a := l.a$
- 2:  $l.a := \max(l'.a, pt.a)$  //tracking maximum time event,  $pt.a$  is physical time at  $a$
- 3: If  $(l.a = l'.a)$  then  $c.a := c.a + 1$  //tracking causality
- 4: Else  $c.a := 0$
- 5: Timestamp event with  $l.a, c.a$

**Receive Event of message  $m$** 

- 6:  $l'.a := l.a$
  - 7:  $l.a := \max(l'.a, l.m, pt.a)$  // $l.m$  is  $l$  value in the timestamp of the message received
  - 8: If  $(l.a = l'.a = l.m)$  then  $c.a := \max(c.a, c.m) + 1$
  - 9: Elseif  $(l.a = l'.a)$  then  $c.a := c.a + 1$
  - 10: Elseif  $(l.a = l.m)$  then  $c.a := c.m + 1$
  - 11: Else  $c.a := 0$
  - 12: Timestamp event with  $l.a, c.a$
- 

A key property of HLC is that it allows us to capture (one-way) causality between two events. If  $e$  happened before ( $hb$ )  $f$  (as defined by Lamport [11]), then  $HLC.e < HLC.f$ , where  $HLC.e < HLC.f$  iff  $(l.e < l.f \vee ((l.e = l.f) \wedge c.e < c.f))$ . This implies that if  $HLC.e = HLC.f$ , then  $e$  and  $f$  are (causally) concurrent. Based on this implication, we use HLC to perform predicate detection. We define a predicate to be  $\epsilon$ -true if it is true for at least  $\epsilon$  time in true physical time. We then say that a monitor is  $\epsilon$ -sensitive if it can detect all  $\epsilon$ -true predicates. It is easy to verify that our monitor algorithms are  $2\epsilon$ -sensitive where  $\epsilon$  is the synchronization error bound. We explain this in more detail in the next section.

### 3. PREDICATE DETECTION FRAMEWORK


Figure 1: Intervals of Process  $P_1$  timestamped using HLC

In this section, we present our general framework for performing predicate detection with the help of HLC. We begin by describing predicate detection for WCPs where our goal is to detect conjunctive predicates across the  $n$  processes, i.e., the predicate is of the form  $\bigwedge_{j=1}^n pr_j$ , where  $pr_j$  is the local predicate at process  $j$ . We then describe how we can extend our framework to detect other forms of predicates.

The fundamental idea underlying our framework is the following. Each process  $j$  reports the intervals when predicate  $pr_j$  is true to the monitor. More specifically, it reports the interval  $[(l_1, c_1); (l_2, c_2)]$  where  $(l_1, c_1)$  is the timestamp of the event when the local predicate first became true and continued to be true until  $(l_2, c_2)$  which is the timestamp of the event when the local predicate became false, i.e. the predicate was true at any timestamp  $t$  s.t.  $(l_1, c_1) \leq t < (l_2, c_2)$ . Note that the interval is left-closed and right-open so it includes the starting timestamp  $(l_1, c_1)$  but does not include the ending timestamp  $(l_2, c_2)$ . Thus, if we have an interval ending at time  $t$  and another interval beginning at time  $t$ , the ending event occurs before the begin event in the sorted list. If multiple endpoints start/end at the same time, their ordering can be arbitrary.

The monitor collects the reported intervals and processes the two corresponding **change points**, the starting timestamp and the ending timestamp, to determine if there is a time  $t$  where  $pr_j$  is true for all  $j$ . Given that we are working with HLC, this reduces to finding a timestamp  $t$  such that  $t$  is included in some interval reported by every process.

We illustrate reporting and timestamping in Figure 1 where for simplicity we set the  $c$  values to be 0. In this figure, process  $P_1$  has two intervals during which its local predicate was true. Process  $P_1$  reports the first interval as  $[(0, 0); (3, 0)]$ , i.e., the local predicate was first true at logical clock value  $(0, 0)$  ( $l = 0, c = 0$ ) and became false at  $(3, 0)$  ( $l = 3, c = 0$ ). Similarly  $P_1$  reports the next interval as  $[(4, 0); (6, 0)]$ .

A potential issue with reporting local predicates as intervals is when the intervals are long. For example, the predicate might always be true for a process or it might be true for a very long time. In such cases, the interval does not get reported till the predicate becomes false which would imply that infinite intervals never get reported. This can also affect the overall latency of the global predicate detection task. To overcome this issue, intervals should be split. That is, we should create an artificial endpoint and report the interval if some threshold is passed before the local predicate becomes false, and simultaneously remember this artificial endpoint as starting point for the next interval. With this

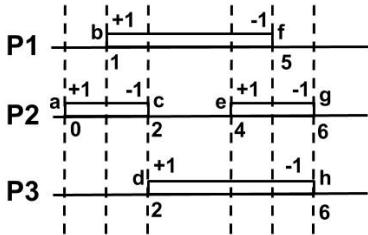


Figure 2: Conjunctive Predicate Detection when Local Predicates are true for some duration of time

assumption, intervals are still reported in entirety. Thus if the predicate is true at time  $t$ , the monitor will report it as a global predicate only after all intervals containing time  $t$  are reported.

We note that it is important that the local processes report the change points as intervals. At first, it might seem fine if the local process reports only the change points individually rather than in pairs as intervals. However, incorrect conclusions can be drawn if the change points arrive to the monitor out of order. For example, in Figure 1, suppose the process reports the four change points individually and they arrive at the monitor in order  $\langle 0, 0 \rangle, \langle 6, 0 \rangle, \langle 3, 0 \rangle, \langle 4, 0 \rangle$ . The monitor will incorrectly assume the predicate holds at this process for the interval  $\langle 3, 0 \rangle, \langle 4, 0 \rangle$  after receiving change point  $\langle 6, 0 \rangle$  until it receives change point  $\langle 3, 0 \rangle$ . The situation becomes even more complex when considering messages from multiple processes. Thus, we require that the local processes report both change points of an interval as a pair to the monitor.

We now briefly describe how our framework can be extended to other predicates. First, it is straightforward to modify our algorithms to handle the case where one is interested in a predicate where  $pr_j$  is true only for a subset (of given size) of processes. A more complex modification is extending our framework to handling arithmetic predicates of the form  $(\sum_{j=1}^n c_j \cdot x_j) \geq C$ , where  $x_j$  is an integer variable of process  $j$ ,  $c_j$  and  $C$  are constants. Here, the change points do not naturally come in pairs and thus we must introduce additional constraints in order to give a correct algorithm. Finally, we can make our algorithms more efficient if we add any assumptions such as messages arrive in FIFO order from a given process or that messages arrive with bounded delay.

## 4. NAÏVE ALGORITHM

In this section, we present our first, naïve, algorithm for detecting WCPs. This algorithm works offline, so we assume that each process reports all of its intervals in sorted order to the monitor when the process terminates. As noted in the previous section, the monitor is searching for a timestamp  $t$  that is included in some interval reported by each of the  $n$  processes. Given this is the goal, we can reduce the problem to finding the maximum overlap among all the intervals. If this overlap is  $n$ , then we know there is a timestamp where the global predicate is true. Moreover, this algorithm immediately extends to the more general case where we want to evaluate whether  $pr_j$  is true for some fixed size subset of processes.

**Outline of the algorithm.** Our algorithm processes each change point one at a time in order from smallest timestamp to largest timestamp maintaining the current overlap

value which is initialized to 0. Recall that each interval has two change points, a left endpoint and a right endpoint. If the current change point is a left endpoint, we add +1 to the current overlap and check if the overlap is  $n$ , in which case we report that the conjunctive predicate is satisfied. If the current change point is a right endpoint, we add -1 to the current overlap. A pseudocode description is shown in Algorithm 2.

---

### Algorithm 2 Naïve Algorithm

---

```

1: loop until there are no unprocessed change points
2:    $t :=$  smallest unprocessed change point timestamp
3:    $X :=$  set of change points with timestamp  $t$ 
4:    $p[t] := 0 //$  net effect at  $t$ , initialized to 0
5:   foreach  $cp \in X$   $\{p[t] := p[t] + p[cp]\}$ 
6:    $overlap\_count := overlap\_count + p[t]$ 
7:   if  $overlap\_count = number\_of\_processes$  then
8:     Report conjunctive predicate detected
9:   end if
10:  end loop

```

---

**Illustration of Algorithm 2.** We illustrate this algorithm using the example in Figure 2 where timestamps are simplified to non-negative integers. Our algorithm first processes change point  $a$  with timestamp 0 (considering  $c = 0$ ). Since  $a$  is a left endpoint,  $overlap\_count$  increases from its initial value of 0 to 1. Our algorithm then processes change point  $b$  with timestamp 1, and  $overlap\_count$  increases to 2. Our algorithm then processes change points  $c$  and  $d$  together with shared timestamp 2. We compute  $p[2] = p[c] + p[d] = 0$  because  $p[c] = +1$  and  $p[d] = -1$ , so  $overlap\_count = 2 + p[2] = 2$ . Our algorithm then processes change point  $e$  with timestamp 3, and  $overlap\_count$  increases to 3. Since this satisfies the condition  $overlap\_count = number\_of\_processes$ , a conjunctive predicate is detected and reported.

**Correctness and Time Complexity.** The correctness follows immediately when we set  $w = n$  from the following elementary Lemma using simple induction.

**LEMMA 1.** *overlap\_count = w at time t if and only if number of intervals that include time t = w*

**THEOREM 1.** *Given a system of n processes and I reported intervals, the running time to find a conjunctive predicate is O(In).*

**PROOF.** In each iteration, we identify and process the unprocessed change point or change points with the lowest timestamp among the  $n$  processes. The cost of this operation is  $O(n)$ . We continue until all change points are processed. The result follows since the number of change points is  $2I$ , two change points per interval.  $\square$

**Advantages and disadvantages of the algorithm.** The naïve algorithm is reasonably efficient with  $O(In)$  worst case complexity and is very easy to understand. However, it has two key disadvantages. First, it requires all events to be available at the beginning. If we try to use this in an online setting where events of different processes are reported at different speeds, then it may not work correctly. For example, using the example of Figure 2, if  $[e, g)$  is reported after  $[d, h)$  so that change point  $d$  is processed after change point  $g$ , then the algorithm will fail to detect that

the global predicate is true at time 4. Second, while its worst case complexity is polynomial, there is a linear dependence on  $n$ , the number of processes. We focus on alleviating these deficiencies in our next algorithm in Section 5.

## 5. RUNTIME DETECTION USING $\mathcal{C}$ -POINT TREE

In Section 4, we described a naïve algorithm with complexity  $O(In)$  that can be used for offline detection of WCP where all the intervals are available in advance. In this section, we focus on adapting that algorithm so that (1) it can be used online, where the monitor receives each interval in a possibly un-sorted manner, and (2) its complexity does not directly depend on  $n$ , the number of processes, though  $I$  typically is larger than  $n$ .

**Change-point tree ( $\mathcal{C}$ -point tree).** Our key step is to use an augmented tree from [5] that we call a change point tree ( $\mathcal{C}$ -point tree) to keep track of the change points reported by processes. The  $\mathcal{C}$ -point tree differs from other augmented trees like interval or segment trees in that nodes represent change points rather than intervals. Specifically, the  $\mathcal{C}$ -point tree maintains the change points in a balanced binary tree sorted by HLC timestamp which allows us to insert or delete change points efficiently. The timestamp of a node is denoted  $t[node]$ . For any  $\mathcal{C}$ -point tree  $T$  with node  $x$ , let  $T(x)$  denote the subtree of  $T$  rooted at node  $x$ . Each node  $x$  is augmented with three additional attributes that allows us to calculate the maximum overlap efficiently. These values are  $p[x]$ , which is  $+1$  or  $-1$  depending on whether  $x$  is a left or right endpoint of an interval,  $v[x]$  which denotes the sum of all  $p$  values of nodes in  $T[x]$ , and  $m[x]$ , the most complex attribute. Let  $w$  be any node in  $T(x)$ ; then  $sum_x(w)$  is the sum of all  $p$  values of nodes in  $T(x)$  with timestamps at most  $t[w]$ . Then  $m[x] = \max_{w \in T(x)} sum_x(w)$ . It follows that, at any time,  $m[root]$  gives the maximum overlap that occurs at any timestamp for the intervals and change points that have been processed.

**Outline of the algorithm.** We now illustrate how the algorithm operates which basically involves how we update the tree when a change point is inserted. We start with an empty  $\mathcal{C}$ -point tree  $T$ . Intervals arrive at the monitor in any order which inserts the left and right endpoints into  $T$ . Let  $x$  be a change point to be inserted into  $T$ . We set  $p[x]$  based on whether  $node$  is a left or right endpoint. We defer setting  $v[x]$  or  $m[x]$  until  $x$  has been inserted. We then insert  $x$  into  $T$ .

After insertion, which may involve rebalancing  $T$ , we must update  $v[y]$  and  $m[y]$  for each node  $y$  whose subtree was modified during the insertion. Because we are working with a balanced binary search tree, this is at most  $O(\log I)$  nodes. We update  $v[y]$  and  $m[y]$  for the affected nodes in a bottom up fashion as follows.

$$v[y] = v[left\_y] + p[y] + v[right\_y]$$

$$m[node] = \max \begin{cases} m[left\_y], \\ \quad //\text{max is in left subtree of } y \\ v[left\_y] + p[y], \\ \quad //\text{max is at } y \\ v[left\_y] + p[y] + m[right\_y], \\ \quad //\text{max is in right subtree of } y \end{cases}$$

For the above formula, we do need to handle the special case where  $y$  has no left child in which case  $m[y]$  is the max

of the second and third cases. After all nodes have been updated, we check  $m[root]$  to determine if the global predicate has been satisfied. For efficiency, if multiple change points have the same timestamp, we maintain a single node for that timestamp. Pseudocode for the algorithm with this optimization is given in Algorithm 3.

---

**Algorithm 3** Algorithm for inserting a change point into a  $\mathcal{C}$ -point tree

---

**Given a new change point from process  $j$ :**

```

1: node  $x := create\_node(j, p\_clock, p[x])$  //if  $x$  is left end-
   point  $p[x] = +1$ ,  $p[x] = -1$  otherwise
2: If node  $x$  (with  $p\_clock$ ) exists in the tree already then
3:   Append  $j$  to process id of existing node,
   Add  $p[x]$  value to existing  $p[x]$ 
   //  $v[x]$  and  $m[x]$  will be updated below
4: Else
5:   insert_node( $x$ ) // maintains balanced tree
6:  $X^* = \{ y \mid \text{the modification or insertion of node } x \text{ caused}$ 
   a change at node  $y$  or some descendant of node  $y \}$ 
   // node  $x$  is in  $X^*$ 
7: for each node  $y$  in  $X^*$  in a bottom up fashion do
8:    $v[y] := v[l] + p[y] + v[r]$ 
   //l is  $y \rightarrow left\_node$ , r is  $y \rightarrow right\_node$ 
   //if l or r does not exist then  $v[l/r] = 0, m[l/r] = 0$ 
9:    $m\_left := m[l]$ 
10:   $m\_node := v[l] + p[y]$ 
11:   $m\_right := v[l] + p[y] + m[r]$ 
12:  temp := max( $m\_node, m\_right$ )
13:  If l exists, then  $m[y] := max(m\_left, temp)$ 
14:  Else  $m[y] = temp$ 
   // $m[y]$  must include one timestamp
15: end for
16: if  $m[root] = n$  then
17:   max_pred_set := trace_back( $m[root]$ )
18: end if

```

---

**Illustration of Algorithm 3.** We revisit the example in Figure 2 to illustrate Algorithm 3. Without loss of generality, suppose the intervals are reported to the monitor in the following order:  $[a, c), [b, f), [d, h), [e, g)$ . After the first interval is reported, two nodes are inserted into the tree,  $(P2, 0, +1)$  for change point  $a$  and  $(P2, 2, -1)$  for change point  $c$ . The resulting tree with  $a$  and  $c$  inserted first is shown in Figure 3 (a).

When  $[b, f)$  is reported to the monitor, it inserts two nodes  $(P1, 1, +1)$  and  $(P1, 5, -1)$  corresponding to  $b$  and  $f$  and updates attributes  $m$  and  $v$  of the nodes affected by the insertion as shown in figure 3 (b). We mark affected nodes by a star to identify nodes that need to recompute  $m$  and  $v$  values. When  $[d, h)$  is reported to the monitor, since the tree has an existing node  $(P2, 2, -1)$  with clock value 2, the monitor updates the node  $(P2, 2, -1)$  to  $((P2, P3), 2, 0)$  (Lines 2 to 3); note the  $p$  value increases from  $-1$  to  $0$ . Then the monitor creates and inserts node  $(P3, 6, -1)$  resulting in the tree shown in Figure 3 (c). Finally, when  $[e, g)$  is reported to the monitor, the insertion of node  $(P3, 4, +1)$  for change point  $e$  will trigger  $m[root]$  to be 3 and the monitor can report that the predicate was satisfied. The final tree after change point  $g$  is processed, which results in existing node  $(P3, 6, -1)$  changing to  $((P2, P3), 6, -2)$ , is shown in Figure 3 (d).

We now prove that this algorithm is sound (if it concludes

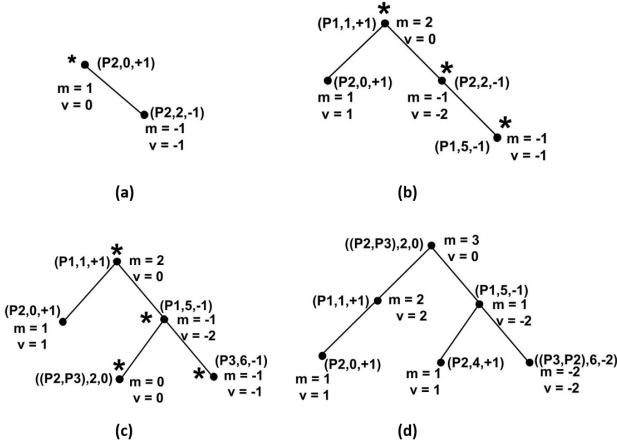


Figure 3: An Illustrative Example for Algorithm 3

the WCP is true, then there exists an HLC timestamp  $t$  such that all  $n$  local predicates are true at  $t$ ) and complete (if there exists an HLC timestamp  $t$  where all  $n$  local predicates are true, then the algorithm will detect it)<sup>1</sup>.

**LEMMA 2.** *Let  $T$  be a C-point tree where for every node  $x \in T$ ,  $v[x] = \text{sum of the } p \text{ values of all nodes in } T(x)$ . Now suppose we execute Algorithm 3 to create a new tree  $T'$  by inserting a new node or modifying an existing node. Then  $T'$  is also a C-point tree where for every node  $x \in T'$ ,  $v'[x] = \text{sum of the } p \text{ values of all nodes in } T'(x)$ .*

**PROOF.** This clearly holds for all nodes not in  $X^*$  since those subtrees have not been modified. We now show it holds for the nodes in  $X^*$ . We process the nodes in  $X^*$  in bottom up fashion, so when we process node  $y$  from  $X^*$  and execute line 8, we know that  $v[l]$  and  $v[r]$  are set correctly, so setting  $v[y] = v[l] + p[y] + v[r]$  makes  $v[y]$  correct.  $\square$

**THEOREM 2.** *Given a system of  $n$  processes and  $I$  reported intervals, the running time to find a conjunctive predicate using a C-point tree is  $O(I \log I)$ .*

**PROOF.** This follows from the fact that processing each change point corresponds to inserting/modifying one entry in a binary search tree and updating the affected  $v$  and  $m$  values. By the properties of balanced binary search trees, the cost of processing each change point is  $O(\log I)$ . Each interval has two change points. Thus, the above lemma follows.  $\square$

**Advantages and disadvantages of the algorithm.** This algorithm has several advantages over the naïve algorithm of the previous section. First, it can be used in an online manner at runtime and is robust against messages arriving out of order and with arbitrarily large delays as long as the delays are finite. Whenever all the necessary intervals have arrived to allow the detection of the global predicate, this will be available at the root node. In fact, once the intervals have been inserted, predicate detection just means

<sup>1</sup>In this claim, we assume that if there are events at timestamp  $t$  and  $t'$ , then we can create an arbitrary number of events in between them with timestamp  $t''$  such that  $t < t'' < t'$ . For example, we can have local events with timestamps  $(l=2, c=0)$  and  $(l=3, c=0)$  between events  $b$  and  $f$  in Figure 2

checking if  $(m[\text{root}]) = n$  which requires  $O(1)$  time. The main drawback with this algorithm is that  $I$  may be large; in fact,  $I \log I$  may be larger than  $I_n$ . Unfortunately, the running time cannot be improved without making some additional assumptions because we must sort the change points when there is no guarantee about the ordering of intervals, and sorting  $I$  elements requires at least  $\Omega(I \log I)$ . We next explore some possible improvements when we make some additional assumptions.

## 6. RUNTIME DETECTION WITH MESSAGING PROPERTIES

In this section, we return to the naïve algorithm from Section 4 and show that we can perform runtime monitoring and reduce worst case time complexity if we add some assumptions about message delivery to the monitor. We consider two types of message delivery assumptions: (1) local ordering property and (2) bounded message delay. We first focus on local ordering property and then show how to adapt this algorithm to leverage bounded message delay.

### 6.1 Local Ordering Property

We first show that we can reduce the worst case time complexity to  $O(I \log n)$  and perform runtime monitoring if we add the following local ordering property for messages:

Intervals reported by a process arrive in a FIFO manner at the monitor.

For example, in Figure 2, we require that the interval  $[a, c)$  is reported to the monitor before interval  $[e, g)$ . However, intervals of different processes may occur in any order. For example interval  $[b, f)$  may be reported before or after interval  $[d, h)$ .

**Outline of the algorithm.** In the naïve algorithm, we processed change points in order of timestamp using a naïve method for identifying the unprocessed change point with earliest timestamp. We improve the algorithm by using a heap to more efficiently identify the change point with the earliest timestamp. Specifically, we maintain a min-heap of size exactly  $n$ , containing exactly one change point per process, where we ensure that change points that are right endpoints are considered to be smaller than change points that are left endpoints if they have the same timestamp. For each process, all other points are easily stored in sorted order because of our assumption that the intervals arrive at the monitor in order.

For both offline and online execution, we initialize the overlap count to 0 and build an initial heap with  $n$  change points, one per process. In online execution, if we do not have one change point per process, we wait until this is the case. For both offline and online execution, whenever we have  $n$  change points, we extract the point with minimum timestamp from the heap and update the current overlap count by +1 or -1 depending on the  $p$  value of the extracted point. If the overlap count reaches  $n$ , we report the WCP is satisfied. Otherwise, we insert the next change point from the same process in the heap. For online execution, if no next change point is available, we wait until the next interval from that process arrives. For offline execution where we have all the points, we continue until all points are consumed. The details of the online algorithm are as shown in Algorithm 4.

**Illustration of Algorithm 4.** Once again, we revisit Figure 2. For sake of discussion, the intervals arrive at

**Algorithm 4** Online min-heap algorithm based on local ordering property.

---

```

1:  $H :=$  min-heap containing first endpoint of each process
2:  $overlap\_count := 0$ 
3: loop
4:    $p := ExtractMin(H)$ 
5:    $overlap\_count := overlap\_count + p.val$ 
6:   if  $overlap\_count = number\_of\_processes$  then
7:     Report conjunctive predicate detected
8:   end if
9:    $Insert(H, \text{next endpoint from } p.process\_id)$ 
10:  // wait if next endpoint is not available yet.
11: end loop

```

---

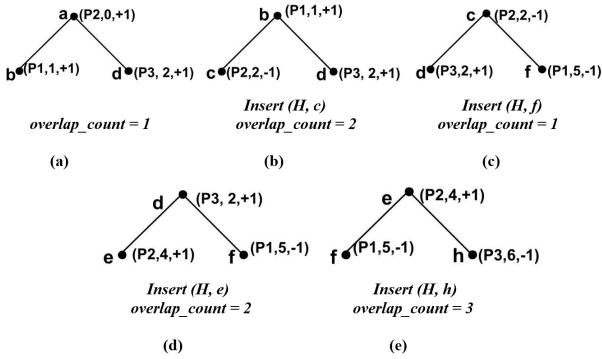


Figure 4: An Illustrative Example for Algorithm 4

the monitor in the following order:  $[a, c), [b, f), [e, g), [d, h)$ . First we fill in the heap of size  $n = 3$  by inserting change points  $a, b$  and  $d$ . Once the heap is full, which does not occur until  $[d, h)$  arrives, we begin extracting the minimum from the heap. In this case, we extract  $a$  and add  $a.val$  to  $overlap\_count$ . Then, we insert  $c$  into the heap as  $a$  was extracted and  $a$  was from process 2. We repeat this process until we detect the conjunctive predicate is satisfied when change point  $e$  is processed as shown in Figure 4. Note that change point  $c$  is processed before  $d$  because  $c$  is a right endpoint. Further note that when running online, once we process the next change point  $f$ , the monitor would wait until process  $P1$  delivers another interval to the monitor.

**Proof of correctness and complexity (sketch).** Observe that Algorithm 4 essentially behaves in the same fashion as the Algorithm 2 except for the use of the heap. It considers the lowest unprocessed event and processes it to calculate the overlap count. Hence, the correctness of the algorithm follows from the correctness of Algorithm 2, our assumption that the intervals arrive to the monitor in order, and that we only extract the minimum from the heap when we have  $n$  points in the heap.

Regarding complexity, observe that the size of the heap is always at most  $n$ . To process each change point, we take  $O(1)$  time to find the minimum,  $O(\log n)$  time to remove this minimum from the heap and  $O(\log n)$  to insert the next change point. Hence, the complexity to process one change point is  $O(\log n)$ . Since the number of change points is  $2I$ , the overall complexity is  $O(I \log n)$ .

**Advantages and disadvantages of this algorithm.** This algorithm has two key advantages: it has  $O(I \log n)$  worst case complexity so it is efficient, and it can be used

both offline and online. It has the disadvantage that it requires intervals from each process to arrive in FIFO order at the monitor. The online version has the disadvantage that if some process takes a long time to report an interval, the monitor must buffer change points reported by other processes. Such a delay may occur because the local event at that process is false or because of message delays.

## 6.2 Bounded Message Delay

In the previous subsection, we assumed that messages from each process were delivered to the monitor in FIFO order. An alternative assumption is to assume that messages are delivered with bounded delay. Specifically, we introduce the following bounded message delay property.

At time  $t$ , the monitor has received any message that was sent by time  $t - \Delta$ .

Observe that the larger  $\Delta$  is, the weaker delivery guarantee property is. Specifically, if there is no assumption on the maximum delay for any message sent to the monitor, then  $\Delta = \infty$ .

**Outline of the algorithm.** We now show how to adapt the previous algorithm that receives intervals in FIFO order to work given the bounded delay assumption. Recall that Algorithm 4 maintains the unprocessed change points for each process in a sorted list and that newly arrived change points are just appended to the end of the appropriate list. We must modify the management of this list given that change points may not arrive in order. We do so as follows.

At any point in time  $t$ , the monitor maintains a latest timestamp value  $max(t)$ . This is initially set to 0. We maintain the invariant that all change points with timestamps at most  $max(t) - \Delta$  have been placed into the sorted lists for their respective processes (and perhaps subsequently entered into the min-heap where points are actually processed). The remaining change points are stored in a single min-heap of unplaced change points where the heap order is still based on timestamp.

When the monitor receives a new message, it checks to see if the message results in an update to  $max(t)$ . If so, it updates  $max(t)$ , adds the two new change points to its min-heap of unplaced change points, and then extracts the minimum timestamp change point  $x$  if that change point has timestamp  $t(x) \leq max(t) - \Delta$  placing  $x$  into its process's sorted list of points. This extraction continues until the minimum timestamp change point  $x$  has timestamp  $t(x) > max(t) - \Delta$ .

The correctness of this algorithm follows from the assumption that messages are delivered with at most  $\Delta$  delay and messages are sent at least every  $\delta$  time units. Specifically, consider a change point  $x$  from process  $j$  with timestamp  $t(x) \leq max(t) - \Delta$ . Any interval from  $j$  that ends before  $t(x)$  must have been sent before  $t(x)$  and thus would be received by time  $max(t)$  given the bounded message delay assumption. Thus, we are sure that change point  $x$  is correctly sorted in the list of change points of process  $j$ ; we do require that right endpoints are placed before left endpoints if they have the same timestamp.

The running time for this algorithm is identical to that of Algorithm 4 except for the min-heap required to process unplaced change points. We observe this min-heap for unplaced points must have bounded size as follows. The timestamp of every change point on a given process is unique. Also, in HLC [10], the timestamp is of the form  $\langle l, c \rangle$ , where

$l$  captures the most recent physical time that the change point was aware of and  $c$  is a bounded counter. In [10], it has been shown that the theoretical maximum value of  $c$  is  $O(n)$ . (In practice, this value remains less than 5.) Hence, the number of change points reported by a process in  $\Delta$  time units is  $O(\Delta n)$ . And, the total change points reported by all processes in  $\Delta$  time units is  $O(\Delta n^2)$ . Thus, the min-heap for unplaced change points has size at most  $O(\Delta n^2)$  and so the maximum time for any insertion or deletion is at most  $O(\log \Delta n)$  giving a total additive cost of  $O(I \log \Delta n)$ .

**Frequent Updates** We do observe that the monitor can still stall if some process  $j$  takes a long time to send its next message which means the monitor will be stuck waiting to receive a message from  $j$  and the min-heap for processing minimum change points cannot advance. Clearly, if this algorithm is intended to be used in an online setting where the monitor should detect satisfaction of the given predicate in a timely manner, we must ensure that the messages not only reach the monitor in a timely manner but that each process must provide frequent updates. Specifically, we must ensure that each process  $j$  sends at least one message to the monitor within each closed interval of length  $\Delta$ . For example, if the predicate value does not change for a long time resulting in a long interval, we require the process to split the interval and send a message within  $\Delta$  of its last sent message. Without this second condition, even with bounded message delay, the monitor cannot be certain that it has up to date information on process  $j$  and cannot detect predicate satisfaction in a timely manner.

**Advantages and disadvantages of this algorithm.** This algorithm has two advantages: it has  $O(I \log \Delta n)$  worst case complexity so it is efficient, and it can be used both offline and online. It has the disadvantage that it requires intervals from each process to arrive at the monitor with bounded delay. The online version has the disadvantage that if some process takes a long time to report an interval, the monitor must buffer change points reported by other processes. Such a delay may occur because the local event at that process is false or because of message delays. This can be overcome if processes send frequent updates.

## 7. ARITHMETIC PREDICATE DETECTION

In this section, we show how the algorithms developed in Sections 4–6 can be adapted for arithmetic predicates. For arithmetic predicates, we assume that each process  $j$  has a variable  $x_j$  and we wish to test some inequality  $f(x_1, \dots, x_n) \geq C$ . The algorithms we present in this section can handle any standard arithmetic function  $f$  such as summation, average, polynomial computation, etc. For illustration purposes, we focus our discussion on arithmetic predicates of the type  $(\sum_{j=1}^n c_j \cdot x_j) \geq C$ , where  $x_j$  is an integer variable of process  $j$ ,  $c_j$  and  $C$  are constants. Detection of such predicates is essential in several real-time distributed systems. For example, suppose we have a distributed system of  $m$  sensors where we would like to detect if the overall pressure sensed by the sensors in the system exceeds  $K$ . The predicate we want to detect is  $(\sum_{j=1}^m c_j \cdot x_j) \geq K$ , i.e.  $c_1 \cdot x_1 + c_2 \cdot x_2 + c_3 \cdot x_3 + \dots + c_m \cdot x_m \geq K$ , where  $x_j$  is pressure sensed by sensor  $j$  and  $c_j$  is the sensitivity factor of sensor  $j$ . These predicates can also be used for tracking constraints like resource usage, geographic distribution of robots, etc.

**Key Characteristic of Arithmetic Predicates.** Conjunctive predicates map obviously into intervals where the predicate is true. However, the same is not true with arith-

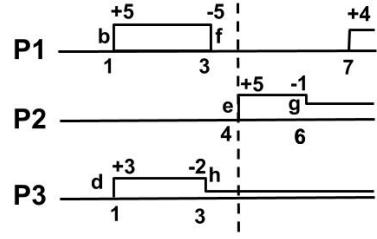


Figure 5: Arithmetic Predicate Detection: False Positive Scenario

metic predicates. For example, the value of  $x_j$  could change from 2 to 3, then back to 1 and so on. Also, the algorithms in Sections 4–6 rely on reporting of intervals in their entirety (with suitable approach for breaking long intervals). This aspect needs to be changed for arithmetic predicates since a process can report individual changes (either every time or in a batch) to the monitor.

We consider two alternatives for reporting change points for arithmetic predicates. The first is to report a single change point at a time where we report the *incval* by which the variable has changed. For example, if the variable increases from 4 to 7 and then to 6, the process reports the change to 7 as an *incval* of 3 with its corresponding timestamp, and then the process reports the change to 6 as an *incval* = -1 with its corresponding timestamp. The second alternative is to impose an interval structure on the change points by having the process report consecutive change points in overlapping pairs. Given the same example where the variable changes from 4 to 7 and then to 6, the process reports the pair of values (4, 7) and their corresponding timestamps when it processes the change to 7. It then reports the pair of values (7, 6) and the corresponding timestamps when it processes the change to 6. If  $x_j$  is intended to have a non-zero value in the initial state, the process reports this initial value with a timestamp of 0 to the monitor. As with interval reporting for conjunctive predicates, if the variable value is unchanged for long periods of time, we may wish to have the processes report the unchanging variable value using the interval reporting structure. We use this structure even in the case where processes typically report a single change point so that the meaning of this message is unambiguous. We further note that this second reporting structure comes with minimal cost as we simply have the process add a report for the previous change point when reporting the new change point.

### 7.1 Offline Arithmetic Predicate Detection

We first observe that in the offline setting where all change points are available up front, we can run any of our other algorithms with at worst an  $O(I \log I)$  additive term by sorting the change points in order of timestamp and then executing the algorithms. This applies regardless of reporting mechanism. If we assume the points from each process are reported in order, then the algorithms can be applied with no additional  $O(I \log I)$  cost. The only subtle point we must watch out for is if multiple change points have the same timestamp. In the case of conjunctive predicates, we handled this by ensuring we processed right endpoints before processing left endpoints when multiple points had the same timestamp. In this case with arithmetic predicates, we must process all

the change points for a given timestamp before we can assert the predicate is true at that timestamp.

## 7.2 Online Arithmetic Predicate Detection

We next observe that in the online scenario, the reporting mechanism is very important.

**LEMMA 3.** *If a process reports only single change points to the monitor and we make no assumptions about message order or message delay, then any online algorithm is either not sound or not complete.*

**PROOF.** Consider the scenario depicted in Figure 5, and suppose that we want to detect the predicate  $x_1 + x_2 + x_3 \geq 10$  and all the variables have the value 0 initially. Note that the variables never sum to 10, so the predicate is never satisfied. Suppose that all the change points except point  $f$  are reported to the monitor. Given the information on hand, the monitor might conclude that the predicate is satisfied at timestamp 4 as the information it has implies  $x_1 = 5$  at time 4. To avoid reporting a false positive and maintain soundness, the online monitor cannot report this predicate being satisfied. However, we can create an almost identical scenario that differs only in that change point  $f$  never occurs. In this second scenario, the predicate is satisfied at timestamp 4. To avoid not reporting a true positive and maintain completeness, the online monitor must eventually report this predicate being satisfied at timestamp 4. Without any assumptions on message order or message delay, the online monitor can never be sure which scenario is the true scenario and the result follows.  $\square$

Given Lemma 3, we must either assume the second reporting mechanism or make some additional assumption about message order or message delivery. We consider both options below.

**Local Ordering Property.** If we assume that each process delivers messages in order to the monitor as in Section 6.1, then we can use either reporting mechanism if we employ Algorithm 4 from Section 6.1. The key observation is that when we process a change point, we know that it is the change point with the minimum timestamp among all processes. As was the case in Section 6.1, the online monitor may stall and require large buffers if some processes are slow to deliver messages. Also, as noted above, we cannot report predicate satisfaction at timestamp  $t$  until all change points with timestamp  $t$  have been processed.

**Bounded Message Delay Property.** If we assume that each process delivers messages with bounded delay to the monitor as in Section 6.2, then we can use either reporting mechanism if we employ the modified algorithm from Section 6.2. To ensure timely predicate detection, we do need to further assume frequent updates from the processes.

**Change Points Delivered in Pairs.** If we assume instead that change points are delivered in pairs to the online monitor, we can again use Algorithm 4 from Section 6.1 with only a slight modification without assuming anything about message order or message delay. This is similar to the modification we made in Section 6.2.

Recall that Algorithm 4 maintains the unprocessed change points for each process in a sorted list and that newly arrived change points are just appended to the end of the appropriate list. We must modify the management of this list given that change points may not arrive in order. We do so as follows. For each process  $j$ , the online monitor stores

$j$ 's unprocessed messages and it maintains  $j$ 's current maximum validated timestamp  $val(j)$ . The idea behind  $val(j)$  is that  $val(j)$  is the largest timestamp such that the monitor knows the exact value of  $x_j$  for all timestamps in the interval  $[0, val(j))$ . We maintain the invariant that all of process  $j$ 's change points with timestamp up to  $val(j)$  are placed into process  $j$ 's sorted list of change points (and perhaps subsequently entered into the min-heap). Initially  $val(j)$  is undefined and we have each process send out a special message with timestamp 0 and the initial value of  $x_j$ .

When the monitor receives a new message from process  $j$ , it checks to see if the first change point in the message has timestamp identical to  $val(j)$  or if it is the special initialization message. If so, then it can update  $val(j)$  and add the corresponding change point into  $j$ 's sorted queue. It then repeatedly checks its already received messages to determine if the next message in order has already been received, in which case it updates  $val(j)$  and adds the next change point into  $j$ 's sorted queue until this cannot continue. Otherwise, the message is added to the unprocessed messages for  $j$ .

The rest of Algorithm 4 is unchanged. The extra time required for processing is the management of the unsorted messages. We can maintain this list using a min-heap which would require at most  $O(\log I)$  time for each insertion and deletion adding, in the worst case, an  $O(I \log I)$  complexity term to the running time of the algorithm. This assumes a worst case where each process's min-heap is large. In practice, this seems unlikely and the algorithm will likely run more quickly.

## 8. COMPARISON OF HLC AND VC FOR PREDICATE DETECTION

In this section, we compare VC based predicate detection and HLC based predicate detection to identify when HLC based predicate detection will be most applicable.

Since VC based algorithm (e.g., [7]) uses vector clocks messages sent by the algorithm have  $O(n)$  size. By contrast, HLC based approach considered in this paper have messages whose size is  $O(1)$ . This is the key advantage of HLC based detection especially if we intend to utilize it in a large system, say with hundreds of processes.

In [7], if the predicate is true in an interval but a message is sent during that interval then that interval needs to be split into two. In other words, it needs to be reported as two different messages. By contrast, in HLC based detection, the interval needs to be reported only once. However, because HLC based detection needs to split long intervals, it will send more messages to the monitor if the underlying system does not send any messages. By contrast, in [7], the message to the monitor needs to be sent when the predicate becomes true. Subsequently as long as there are no messages in the underlying system, no messages need to be sent to the monitor. Given these two issues, we can expect that the number of messages  $m$  sent by the underlying system (when the local predicate at the sender was true) would be roughly equal to the number of intervals ( $I$ ) reported by the HLC algorithm.

We compare the complexity of our algorithm with that of [7] under the assumption that  $m \approx I$ . Number of messages sent to the monitor is  $O(m)$  in [7] whereas the number of messages sent to the monitor is  $O(I)$  with HLC. However, the message size in  $O(n)$  in [7]. By contrast, the message size is  $O(1)$  with HLC. The total work performed by the monitor in [7] is  $\Theta(mn)$ . By contrast, the complexity of

HLC based detection is  $O(I \log I)$  and can be reduced to  $O(I \log n)$  under suitable assumptions.

A related question is whether we could have used the standard logical clocks (LCs) [11] to achieve the same results in this paper. While the algorithms in the paper could be used with LCs and there is no change in the complexity or correctness, using LCs will have overwhelming false-negative ratio as LC will miss almost all occurrences of consistent states. In particular, because LC does not incorporate physical time and makes no synchronization assumption, LC monitoring is not  $2\epsilon$ -sensitive and may fail to identify a global predicate that is true for an extremely long period of time.

## 9. CONCLUSION

In this paper, we focused on the problem of detecting predicates in a partially synchronous system. In these systems, applications often rely on assumptions provided by the underlying system to perform their computation. When these assumptions do not match with the actual guarantees provided by the system, there is a potential for false positives (finding non-existent bugs) and false negatives (missing bugs). In these systems, (hybrid) vector clocks (HVCs) and (hybrid) logical clocks (HLCs) provide solutions at the two extremes: The former ensures no false negatives but permits false positives whereas the latter ensures no false positives but permits false negatives. In the case of HLC, this false negative rate is low when we are working with predicates that are stable for a duration and the rate of false negatives decrease as the duration increases.

We presented several algorithms for detecting predicates in a distributed system. Of these, the algorithm in Section 5 makes the least assumptions and provides a complexity of  $O(I \log I)$ . It allows communication with the monitor to be non-FIFO and have an arbitrary delay. The algorithm in Section 6.1 reduces the complexity to  $O(I \log n)$  under the assumption that the monitor receives the messages in a FIFO manner. The algorithm in Section 6.2 obtains a complexity of  $O(I \log \Delta n)$  where  $\Delta$  captures the window within which messages sent to the monitor are received. We also extended these algorithms for detecting arithmetic predicates, which provide a general purpose framework for detecting predicates associated with resource usage in a distributed system, geographic distribution of robots in a field, etc.

## 10. ACKNOWLEDGMENTS

This work is partially supported by NSF CNS 1329807, NSF CNS 1318678, and XPS 1533802.

## 11. REFERENCES

- [1] R. Atreya, N. Mittal, A. D. Kshemkalyani, V. K. Garg, and M. Singhal. Efficient detection of a locally stable predicate in a distributed system. *Journal of Parallel and Distributed Computing*, 67(4):369–385, 2007.
- [2] C. M. Chase and V. K. Garg. Detection of global predicates: Techniques and their limitations. *Distributed Computing*, 11(4):191–201, 1998.
- [3] M. Chow, D. Meisner, J. Flinn, D. Peek, and T. Wenisch. The mystery machine: End-to-end performance analysis of large-scale internet services. In *Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 217–231, 2014.
- [4] J. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, W. Hsieh, S. Kanthak, E. Kogan, H. Li, A. Lloyd, S. Melnik, D. Mwaura, D. Nagle, S. Quinlan, R. Rao, L. Rolig, Y. Saito, M. Szymaniak, C. Taylor, R. Wang, and D. Woodford. Spanner: Google’s globally-distributed database. In *Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 251–264, 2012.
- [5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms* (3. ed.). MIT Press, 2009.
- [6] C. J. Fidge. Timestamps in message-passing systems that preserve the partial ordering. In *Proceedings of the 11th Australian Computer Science Conference (ACSC)*, pages 56–66, 1988.
- [7] V. K. Garg and C. M. Chase. Distributed algorithms for detecting conjunctive predicates. In *Proceedings of the 15th International Conference on Distributed Computing Systems (ICDCS)*, pages 423–430, 1995.
- [8] V. K. Garg, C. M. Chase, J. R. Mitchell, and R. B. Kilgore. Detecting conjunctive channel predicates in a distributed programming environment. In *Proceedings of the 28th Annual Hawaii International Conference on System Sciences (HICSS)*, pages 232–241, 1995.
- [9] V. K. Garg and B. Waldecker. Detection of weak unstable predicates in distributed programs. *IEEE Transactions on Parallel and Distributed Systems*, 5(3):299–307, 1994.
- [10] S. S. Kulkarni, M. Demirbas, D. Madappa, B. Avva, and M. Leone. Logical physical clocks. In *OPODIS*, pages 17–32, 2014.
- [11] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, July 1978.
- [12] F. Mattern. Virtual time and global states of distributed systems. In *Parallel and Distributed Algorithms*, pages 215–226. North-Holland, 1989.
- [13] D. Mills. A brief history of ntp time: Memoirs of an internet timekeeper. *ACM SIGCOMM Computer Communication Review*, 33(2):9–21, 2003.
- [14] K. Sen, G. Rosu, and G. Agha. *Online Efficient Predictive Safety Analysis of Multithreaded Programs*, pages 123–138. Springer Berlin Heidelberg, 2004.
- [15] B. Sigelman, L. Barroso, M. Burrows, P. Stephenson, M. Plakal, D. Beaver, S. Jaspan, and C. Shanbhag. Dapper, a large-scale distributed systems tracing infrastructure. Technical report, Google, Inc., 2010.
- [16] Twitter zipkin. <https://github.com/twitter/zipkin>. [Online; accessed 31-October-2016].
- [17] S. Yingchareonthawornchai, S. S. Kulkarni, and M. Demirbas. Analysis of bounds on hybrid vector clocks. In *OPODIS*, pages 34:1–34:17, 2015.
- [18] S. Yingchareonthawornchai, D. N. Nguyen, V. T. Valapil, S. S. Kulkarni, and M. Demirbas. Precision, recall, and sensitivity of monitoring partially synchronous distributed systems. In *Proceedings of the 16th International Conference on Runtime Verification (RV)*, pages 420–435, Madrid, Spain, 2016. Extension available at arXiv:1607.03369.

# Accountability in Dynamic Networks

Xavier Vilaca

INESC-ID, Instituto Superior Técnico,  
Universidade de Lisboa  
xvilaca@gsd.inesc-id.pt

Luís Rodrigues

INESC-ID, Instituto Superior Técnico,  
Universidade de Lisboa  
ler@tecnico.ulisboa.pt

## ABSTRACT

A variety of distributed protocols require pairs of neighbouring nodes of a network to repeatedly interact in pairwise exchanges of messages of mutual interest. Well known examples include file-sharing systems [4] and gossip dissemination protocols [18, 17], among many other examples. These protocols can operate in very diverse settings such as wireless ad-hoc or peer-to-peer overlay networks. These settings pose three main challenges. First, networks are inherently dynamic, whether due to uncontrolled mobility or maintenance of the overlay. Second, since communication may be costly, nodes may act rationally by not sending messages, while still receiving messages from their neighbours. Third, nodes may have incomplete information about the network topology. In this work, we aim at gaining theoretical insight into how to persuade agents to exchange messages in dynamic networks.

We take a game theoretical approach to determine necessary and sufficient conditions under which we can persuade rational agents to exchange messages, by holding them accountable for deviations with punishments. Unlike previous work [19, 6, 7, 5, 20, 18, 17, 9], we do not assume the network is the result of rational behaviour or that the topology changes according to some known probability distribution, and we do not limit our analysis to one-shot pairwise interactions. We make three contributions: (1) we provide a new game theoretical model of repeated interactions in dynamic networks, where agents have incomplete information of the topology, (2) we define a new solution concept for this model, and (3) we identify necessary and sufficient conditions for enforcing accountability, i.e., for persuading agents to exchange messages in the aforementioned model.

Our results are of technical interest but also of practical relevance. We show that we cannot enforce accountability if the dynamic network does not allow for *timely punishments*. In practice, this means for instance that we cannot enforce accountability in some networks formed in file-sharing applications such as BitTorrent [4]. We also show

that for applications such as secret exchange, where the benefits of the exchanges significantly surpass the communication costs, timely punishments are enough to enforce accountability. However, we cannot in general enforce accountability if agents do not possess enough information about the network topology. Nevertheless, we can enforce accountability in a wide variety of networks that satisfy 1-connectivity [15] with minimal knowledge about the network topology. This result provides a sound game theoretical foundation for the empirical results presented in [18, 17, 9], showing that we can in fact enforce accountability in gossip dissemination with connected overlay networks where nodes possess sufficient information about the network.

## 1. INTRODUCTION

A variety of protocols require pairs of neighbouring nodes of a network to repeatedly interact in pairwise exchanges of messages that are of mutual interest to both parties. Well known examples include file-sharing systems [4] and gossip dissemination protocols [18, 17, 9]. These protocols can operate in very diverse settings such as wireless ad-hoc or peer-to-peer overlay networks. These settings pose three main challenges. First, networks are inherently dynamic, whether due to uncontrolled mobility or maintenance of the overlay. Second, since communication may be costly, nodes may act rationally by not sending messages, while still receiving messages from their neighbours. Third, nodes may have incomplete information about the network topology. In this work, we aim at gaining theoretical insight into how to persuade agents to exchange messages in dynamic networks.

A growing body of literature has taken a game theoretical approach [22] to address rational behaviour in a variety of distributed problems in static networks (e.g., [10, 1, 2, 3, 26]). Nodes are viewed as being under the control of rational agents<sup>1</sup> that seek to maximize individual utilities. In these lines, we use Game Theory to address the problem of rational behaviour in pairwise exchanges of messages over links of a dynamic network. We consider that agents obtain a benefit of receiving messages from their neighbours, and incur costs for sending and receiving messages. We are interested in protocols that satisfy two properties: (1) agents exchange messages in every pairwise interaction with neighbours, in order to provide some useful service such as file-sharing and (2) the protocols are equilibria according to some solution concept (e.g., Nash equilibrium), so that no agent gains (increases its utility) by deviating from the protocol. Protocols

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

ICDCN '17, January 04-07, 2017, Hyderabad, India

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007750>

<sup>1</sup>Henceforth, we use the designation agent when referring to both the node and the rational agents.

that satisfy these properties are said to *enforce accountability* in pairwise exchanges.

*Folk Theorems.* Results known as Folk Theorems have shown that it is possible to enforce accountability by holding agents accountable for deviations with punishments that decrease their utility [19], provided that (1) agents want to participate in exchanges, i.e., the benefits of receiving messages outweigh the communication costs, (2) agents perceive interactions to occur infinitely often, which is possible if the end-horizon of interactions is unknown [22], and (3) a monitoring infrastructure provides agents with sufficient information regarding the past behaviour of other agents, so that any deviation from the protocol may be promptly detected and punished. In a distributed system, agents may only learn about past behaviour of other agents in messages received from their neighbours. Therefore, a monitoring infrastructure is constrained by the network. Unfortunately, most proofs of Folk Theorems assume an exogenous monitoring infrastructure, ignoring the constraints imposed by dynamic networks. Some proofs of Folk Theorems take into account network constraints [6, 24], but they assume that networks are static or that agents know a probability distribution over topologies at each point in time. Such knowledge may not be readily available to agents in our setting. As we shall see, this poses multiple challenges not addressed by existing proofs of Folk Theorems.

*Contributions.* We bridge the gap between existing proofs of Folk Theorems and the goal of enforcing accountability in dynamic networks. We make three contributions: (1) we provide a new game theoretical model of repeated interactions in dynamic networks, where agents have incomplete information of the topology, (2) we define a new solution concept for this model, and (3) we identify necessary and sufficient conditions for enforcing accountability in the aforementioned model.

*Game Theoretical Model.* We consider that an *adversary* selects the network topologies [15]. We focus on networks that are not under the control of the agents, exemplified by wireless ad-hoc networks and overlays such as [18, 17]. To capture such exogenous restrictions on the network, we assume that the adversary is oblivious to the messages sent by agents, selecting the topologies for all times prior to the beginning of the exchanges. At each point in time, agents learn partial information about the current topology, including the identities of their current neighbours. They also possess information about the subset  $\mathcal{G}^*$  of dynamic networks that the adversary may generate. In practice, the set  $\mathcal{G}^*$  represents basic information known by agents regarding the network structure. For instance, in an overlay network designed to disseminate data in a reliable fashion, agents expect the adversary to only generate connected networks (hence,  $\mathcal{G}^*$  only contains connected networks), whereas in a more dynamic setting such as a wireless ad-hoc network,  $\mathcal{G}^*$  may contain a wider variety of non-connected networks.

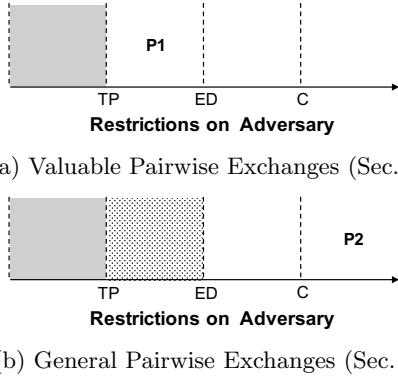
*Solution Concept.* We define a solution concept for our model, named  $\mathcal{G}^*$ -Oblivious Adversary Perfect Equilibrium ( $\mathcal{G}^*$ -OAPE). To understand its definition, it is useful to recall the notion of Nash equilibrium, which states that a protocol is an equilibrium if it is a best response, i.e., no agent can gain by deviating given that other agents also do not deviate. This definition does not suit our purposes for two reasons. First, it does not consider deviations *off the*

*equilibrium path*, i.e., deviations that occur after histories of messages where some agents have deviated [22]. Thus, a Nash equilibrium may rely on empty threats of punishments such as punishments that also cause some utility loss to the punishers. Second, agents must know the probability distribution of the adversary generating each topology. The first issue is addressed in the literature by the notion of sequential equilibrium (SE) [14], which requires protocols to be best responses after every history of messages. To address the second issue, the notion of  $\mathcal{G}^*$ -OAPE refines SE by requiring protocols to be best responses after every history, for any fixed network in  $\mathcal{G}^*$  generated by the adversary. In other words, we require that agents cannot gain from deviating given that the adversary may generate any network in  $\mathcal{G}^*$ , even if the agents know the network generated by the adversary.

*Necessary and Sufficient Conditions.* The main goal of the paper is to determine the degree of dynamism that we can tolerate in order to enforce accountability in pairwise exchanges, i.e., we aim to determine necessary and sufficient restrictions on  $\mathcal{G}^*$  for  $\mathcal{G}^*$ -OAPE protocols that enforce accountability to exist. Our results show that the ability to enforce accountability depends not only on  $\mathcal{G}^*$  but also on the structure and utility of pairwise exchanges.

We determined the weakest set of restrictions on  $\mathcal{G}^*$  for which we can enforce accountability. Our first result shows that  $\mathcal{G}^*$  must satisfy a property named *timely punishments*. Intuitively, this property states that for ever pairwise exchange between agents  $i$  and  $j$ , agent  $j$  must always be able to communicate a deviation of  $i$  to some third agent that later interacts with  $i$ , and thus is able to punish  $i$  for that deviation. This restriction is not met by some networks such as file-sharing overlays (e.g., BitTorrent [4]), where users with similar interests interact frequently with each other but only rarely with users with different interests. Our next result provides a  $\mathcal{G}^*$ -OAPE protocol that enforces accountability, assuming that  $\mathcal{G}^*$  is restricted by timely punishments and that exchanges are *valuable*, that is, agents have a high benefit/cost ratio of receiving/ sending messages and neglect download costs. Such type of exchanges occurs, for instance, when agents share small but highly valuable secrets such as private keys [18, 17].

In many cases, the assumption that pairwise exchanges are valuable is too restrictive. For instance, in file-sharing, we may expect agents to be interested in exchanging large files, but the cost of uploading such files is certainly non-negligible, and the benefit-to-cost ratio of receiving/ sending files may be small. It is therefore of practical interest to understand whether we can enforce accountability in more general settings. Our final two results identify a necessary and a sufficient condition to achieve this goal. Namely, we show that  $\mathcal{G}^*$  must satisfy, in addition to timely punishments, a property that we call *eventual distinguishability*. Roughly speaking, this property states that whenever two (or more) agents may punish  $i$  from defecting in the past (towards some other agent  $j$ ), they have the information necessary to coordinate their actions, so that the total number of additional punishments for a single deviation of  $i$  is never too large. Whether  $\mathcal{G}^*$  satisfies eventual distinguishability depends both on the properties of graphs in  $\mathcal{G}^*$  and on the information available to agents about the topology. Following previous work in dynamic networks [15], we consider a class of connected networks where agents have knowledge



**Figure 1: timely punishments (TP), eventual distinguishability (ED), and connectivity (C).**

of the degree of their neighbours, which are formed by overlay networks such as [18, 17, 9]. We show that if networks in  $\mathcal{G}^*$  are contained in this class of connected networks, then there is a protocol that enforces accountability in general pairwise exchanges.

*Summary of Results.* Our results are summarized in Fig. 1. We show that, in general, timely punishments (grey area) and eventual distinguishability (dotted area) are necessary conditions for enforcing accountability, whereas connectivity is a sufficient condition (protocol  $P1$ ). In valuable pairwise exchanges, timely punishments are necessary and sufficient for enforcing accountability (protocol  $P2$ ).

*Related work.* Existing works have used Game Theory to analyse interactions in file sharing [12, 21, 16] and gossip dissemination [18, 17]. Unlike these works, we do not limit our analysis to one-shot pairwise interactions. Our proposed model and results are more related to work in models of dynamic games and network formation games. In the former, the structure of the game being repeated varies in each repetition according to a known probability distribution [19, 6, 7]. This captures repetitions of the same game where only the network topology varies. In the latter, the network is the outcome of the actions of rational agents [5, 20]. To the best of our knowledge, none of these models captures network variations as being caused by an adversary oblivious to the actions of agents, and they do not model incomplete information of the network topology. Hence, they are not appropriate for modelling unpredictable changes in the network such as physical topology changes. More importantly, they do not address the challenges of devising distributed monitoring mechanisms. Work in reputation systems such as [13, 8] have proposed and analysed distributed systems that perform monitoring. These works are not comparable to ours because they do not prove Folk Theorems.

## 2. MODEL

We consider a synchronous message passing system with reliable communication in a dynamic network. The system entities are the rational agents, which send messages over links of the network, and an oblivious adversary, which selects the dynamic network at the beginning. Specifically, time is divided into rounds. Prior to the execution of the protocol, the adversary selects an evolving graph  $G$  that specifies the communication graph  $G^m$  of each round  $m$ . Let

$\mathcal{G}$  be the set of all evolving graphs, and let  $\mathcal{N} = \{1 \dots n\}$  be the set of agents, where  $n$  is the number of agents. As in [1, 2], we assume that edges are private: agent  $i$  can send a message to  $j$  at round  $m$  only if  $j$  is a neighbour of  $i$  in  $G^m$ , and if  $i$  sends that message, then exactly  $j$  receives it; if  $i$  has multiple neighbours, then  $i$  can discriminate neighbours by sending different messages to each. We also assume for simplicity that graphs are undirected. Finally, we assume that each agent knows  $n$  and the identities of its neighbours in each round  $m$ , prior to sending messages in  $m$ .

### 2.1 Pairwise Exchanges

We consider an infinitely repeated game of pairwise exchanges of messages between neighbouring agents. Given  $G \in \mathcal{G}$ , at each round  $m$ , every two neighbouring agents  $i$  and  $j$  have values of interest to each other. The goal is to persuade  $i$  and  $j$  to share these values plus some additional information required to monitor other agents.

*Actions and histories.* Agents  $i$  and  $j$  may exchange messages in one or more communication steps. For simplicity, we abstract communication by considering a finite set of individual actions, which capture most exchanges of interest, namely (1) *defection*, where  $i$  omits messages (and thus passively punishes  $j$ ), (2) *cooperation*, where  $i$  sends its value plus monitoring information to  $j$ , and (3) *active punishment*, where  $i$  sends messages while causing a utility loss to  $j$  (for instance, if  $i$  sends garbage instead of the value [18]). Our results can easily be generalized to arbitrary (finite) sets of individual actions. In some of our results, we also consider that a punishment can be proportional to some constant  $c$ , i.e., can cause a utility loss proportional to  $c$ ; we also consider a fourth action of *punishment avoidance*, where  $i$  avoids the cost of a punishment but does not receive the value from  $j$ . Both punishment avoidance and proportional punishments are possible actions when multiple communication steps occur between agents; later, we discuss possible implementations.

At every round  $m$ ,  $i$  and  $j$  simultaneously follow an individual action, and are only informed of each other's individual actions at the end of the round. A round- $m$  *action*  $a_i$  of  $i$  specifies the individual actions of  $i$  towards every neighbour. A round- $m$  *action profile*  $\vec{a}$  specifies the round- $m$  actions followed by all agents. When the adversary generates an evolving graph  $G \in \mathcal{G}$ , repeated pairwise exchanges are characterized by a set  $\mathcal{H}(G)$  of *histories*. A round- $m$  history  $h \in \mathcal{H}(G)$  is a pair  $((\vec{a}^{m'})_{m' < m}, G)$  representing the sequence of action profiles  $\vec{a}^{m'}$  followed in rounds  $m' < m$  and the evolving graph  $G$  selected by the adversary. In other words,  $h$  represents global information available immediately before agents follow round- $m$  actions. A *run*  $r$  is a function mapping each round  $m$  to a round- $m$  history  $r(m)$ <sup>2</sup>.

*Information and Strategies.* Information available to agents can be divided into knowledge about the game structure and private observations.

Regarding the former, we assume that the main information about the game structure is common knowledge<sup>3</sup>. In Theorem 1 we show that some minimal restrictions on the evolving graphs generated by the adversary are required

<sup>2</sup>For every  $m' < m$ , the sequence of actions in history  $r(m')$  is a prefix of the sequence of actions in  $r(m)$ .

<sup>3</sup>Every agent knows this, knows that every agent knows this, and so on.

to enforce accountability in dynamic networks. Therefore, the adversary must be constrained to generate only a subset  $\mathcal{G}^* \subseteq \mathcal{G}$ . We also assume that  $\mathcal{G}^*$  is common knowledge. Note that our model is general, and we may consider  $\mathcal{G}^* = \mathcal{G}$  (although, as we show, in this case, it is not possible to persuade agents to exchange messages).

Regarding private observations, given  $G \in \mathcal{G}$  and round  $m$ , every agent  $i$  acquires some information about  $G^m$  prior to sending messages in  $m$ , which includes the identity of the round- $m$  neighbours of  $i$ . We represent this information as a set  $\mathcal{G}_i^m$  of graphs, such that  $G^m \in \mathcal{G}_i^m$  and, in every graph  $\bar{G}$  in  $\mathcal{G}_i^m$  agent  $i$  obtains the same information about  $\bar{G}^m$  and  $G^m$ . For instance, if agents only know the identity of their neighbours, then  $\mathcal{G}_i^m$  is the set of graphs where  $i$  has the same set of neighbours as in  $G^m$ .

Given a round- $m$  history  $h$ , there is a private history  $h_i$  of observations made by  $i$  up to the beginning of round  $m$ , which include the sets  $\mathcal{G}_i^{m'}$  for every round  $m' \leq m$ , the actions of  $i$  in every round  $m' < m$ , and the individual actions followed by neighbours of  $i$  towards  $i$  in every round  $m' < m$ . We can associate to each private history  $h_i$  a round- $m$  information set  $I_i$  containing the histories that provide the same information as in  $h_i$ . Given  $G \in \mathcal{G}$ , we denote by  $\mathcal{I}_i(G)$  the set of information sets of  $i$  compatible with  $G$ . We make the standard assumption that agents have perfect recall, that is, agents recall all their observations made in the past.

A strategy  $\sigma_i$  of agent  $i$  corresponds to a distributed protocol. Specifically, for each  $G \in \mathcal{G}$  and information set  $I_i \in \mathcal{I}_i(G)$ ,  $\sigma_i(\cdot | I_i)$  is a probability distribution over round- $m$  actions available to  $i$ . We use the designation *protocol* for strategy profiles  $\vec{\sigma}$ , which specify the strategies followed by every agent.

*Utility.* When agents follow an action profile  $\vec{a}$ , agent  $i$  obtains a utility  $u_i(\vec{a})$ . This is the sum of the utilities obtained in interactions with each neighbour  $j$  given as the difference between the benefits of receiving a value from  $j$  and the costs of punishments and communication. Specifically, whenever  $j$  follows a cooperation or punishment action and  $i$  does not avoid a punishment,  $i$  obtains a benefit  $\beta$  of receiving  $j$ 's value. Regarding communication costs,  $i$  incurs a normalized cost 1 of sending messages in cooperation and punishment actions, incurs no cost by defecting and avoiding punishments, and incurs the cost  $\alpha$  of receiving messages from  $j$ . Finally, a punishment of  $j$  proportional to  $c$  causes a utility loss of  $c\pi$  to  $i$ , where  $\pi$  is the unitary cost per punishment.

We now define the (total) expected utility  $u_i(\vec{\sigma} | G, I_i)$  of  $i$  when agents follow a given protocol  $\vec{\sigma}$ , conditional on some  $G \in \mathcal{G}^*$  and on a round- $m$  information set  $I_i$ . Given a round- $m$  history  $h \in I_i$ ,  $\vec{\sigma}$  defines a probability distribution over runs compatible with  $G$  and  $h$ . Therefore, we can compute the expected utility of  $i$  in every future round  $m' \geq m$  conditional on  $G$  and  $h$  as the expected value of  $u_i(\vec{a})$ , where the expectation is taken relative to the probability defined by  $\vec{\sigma}$  of agents following action profile  $\vec{a}$  in round  $m'$ . With this in mind,  $u_i(\vec{\sigma} | G, I_i)$  is the expected value of  $u_i(\vec{a})$  for every future round  $m' \geq m$ , as computed in round  $m$ . Formally, we need to take into account both (1) the probability of each  $h \in I_i$  being realized and (2) the effect of time on the value of future utilities. Regarding (1), we consider a belief system  $\mu$ , specifying the probability  $\mu(h | G, I_i)$  that  $h$  is realized. Regarding (2), we follow the standard approach [19] of assuming that the value of future

utilities decays over time with a discount factor  $\delta \in (0, 1)$ , so the expected utility in round  $m'$  discounted to round  $m$  is  $\delta^{m'-m} u_i(\vec{a})$ . Let  $u_i(\vec{\sigma} | G, h)$  be the sum of the expected value of  $u_i(\vec{a})$  for every future round  $m' \geq m$ , discounted to  $m$  by  $\delta$ , conditional on  $G$  and  $h$ ; then,  $u_i(\vec{\sigma} | G, I_i)$  is the expected value of  $u_i(\vec{\sigma} | G, h)$ , where the expectation is taken relative to  $\mu(h | G, I_i)$ .

## 2.2 Enforcing Accountability

We say that a protocol  $\vec{\sigma}$  enforces accountability iff (1) in  $\vec{\sigma}$ , agents always cooperate by exchanging their values, until some agent deviates, and (2)  $\vec{\sigma}$  is an equilibrium, so that no agent gains by deviating. Regarding (2), we need to define a *solution concept*, specifying exact conditions under which  $\vec{\sigma}$  is an equilibrium.

*Solution Concept.* We define a new solution concept for our model, which is a refinement of sequential equilibrium (SE) [14]. In its original definition, a protocol  $\vec{\sigma}^*$  is said to be a SE if there is a belief system  $\mu^*$  consistent with  $\vec{\sigma}^*$  (see below) such that for every agent  $i$  and information set  $I_i$ ,  $i$  cannot gain by deviating from  $\vec{\sigma}^*$  conditioning on  $I_i$ . We refine this definition by also conditioning on every evolving graph  $G \in \mathcal{G}^*$  that may be selected by the adversary. Given this refinement, if  $\vec{\sigma}^*$  is an equilibrium under our solution concept, then no agent  $i$  gains by deviating from  $\sigma_i^*$  given that other agents do not deviate, even if  $i$  knows what is the evolving graph generated by the adversary. In particular, agent  $i$  does not gain from not punishing other agents or not forwarding monitoring information that may trigger additional punishments.

Formally, we say that  $\vec{\sigma}^*$  is a  $\mathcal{G}^*$ -Oblivious Adversary Perfect Equilibrium ( $\mathcal{G}^*$ -OAPE) iff there is a belief system  $\mu^*$  consistent with  $\vec{\sigma}^*$  and  $\mathcal{G}^*$  such that, for every  $G \in \mathcal{G}^*$ , agent  $i$ , strategy  $\sigma_i$ , and information set  $I_i \in \mathcal{I}_i(G)$ ,  $u_i(\vec{\sigma}^* | G, I_i) \geq u_i((\sigma_i, \vec{\sigma}_{-i}^*) | G, I_i)$ , where the expectation is taken relative to  $\mu^*$ , and  $(\sigma_i, \vec{\sigma}_{-i}^*)$  is the protocol where only  $i$  deviates from  $\sigma_i^*$  by following  $\sigma_i$ . The definition of belief system consistent with  $\vec{\sigma}^*$  and  $\mathcal{G}^*$  is very technical, and we leave it to the full paper [23]. Intuitively, given  $G$ , if  $I_i$  is consistent with agents following  $\vec{\sigma}^*$  (i.e., agents could have taken the actions observed in  $I_i$  if they were following  $\vec{\sigma}^*$  and the evolving graph being  $G$ ), then  $i$  believes that agents followed  $\vec{\sigma}^*$ , otherwise,  $i$  believes that they followed some alternative protocol.

## 3. KEY CONCEPTS

In the proofs of our results, we identify multiple key concepts related to the properties of the protocols and evolving graphs. We summarize them here for future reference.

*Safe-bounded protocols.* It turns out that it is relevant to our results to distinguish between safe and non-safe protocols, and between bounded and non-bounded protocols. Regarding the former, a protocol  $\vec{\sigma}$  is *safe* if in every interaction agents either cooperate or punish each other, thus never omitting messages. Otherwise,  $\vec{\sigma}$  is *non-safe*. In the full paper, we show that in general non-safe protocols are not  $\mathcal{G}^*$ -OAPE. Intuitively, when an agent  $i$  defects another as a punishment for a past deviation, that defection can be interpreted as a profitable deviation, i.e., a deviation where  $i$  avoids the cost of forwarding data. As a result, further punishments are triggered to punish  $i$ , and  $i$  gains from not carrying the punishment. This precludes the protocol from

being an equilibrium. For this reason, part of our results focus on incentives that use only active punishments, which cannot be mistaken for profitable deviations. Regarding the latter, a protocol  $\vec{\sigma}$  is *bounded* if the duration of punishments is bounded. Specifically, every protocol can be represented as a state machine [22]; in a bounded protocol, (1) the number of states is finite, (2) some of those states are cooperation states where all agents cooperate, and (3) starting from an arbitrary state, the time of convergence to a cooperation state is bounded. Bounded protocols are simpler to analyse and more useful in practice, since they ensure that memory is bounded, which is an important requirement in dynamic networks. For these reasons, we also restrict part of the analysis to bounded protocols. The full paper also generalizes our results for non-bounded protocols.

*Punishment Opportunities.* Roughly speaking, a punishment opportunity (PO) for an interaction of agent  $i$  with  $j$  is a later interaction between an agent  $l$  and  $i$  where  $l$  may have been informed of a deviation of  $i$  towards  $j$  and thus has the opportunity to punish  $i$ . This requires the existence of a temporal path (a sequence of causally influenced interactions) from  $j$  to  $l$  in the evolving graph such that  $i$  cannot interfere with information forwarded from  $j$  to  $l$ . Formally, given  $G \in \mathcal{G}^*$  and agent  $i$ , a round- $m$   $i$ -edge is a pair  $(j, m)$  where  $(i, j)$  is an edge in  $G^m$ . We say that  $j$  causally influences  $l$  in  $G$  between  $m$  and  $m'$  [15], denoted  $(j, m) \sim_i^G (l, m')$ , if  $m < m'$  and either  $j = l$  or there is a  $j$ -edge  $(o, m'')$  in  $G$  such that  $(o, m'') \sim_i^G (l, m')$ . We say that  $j$  causally influences  $l$  in  $G$  without interference from  $i$  between rounds  $m$  and  $m'$ , denoted as  $(j, m) \sim_i^G (l, m')$ , if the above holds for  $o \neq i$ . A PO of  $i$  for  $(j, m)$  in  $G$  is an  $i$ -edge  $(l, m')$  such that  $(j, m) \sim_i^G (l, m')$ .

*Evasive strategies.* Given a protocol  $\vec{\sigma}^*$ , an evasive strategy  $\sigma'_i$  for agent  $i$  is a strategy where  $i$  first deviates from  $\sigma_i^*$  by defecting some neighbours, and then hides this deviation from as many agents as possible, for as long as possible. In particular,  $i$  may defect a neighbour  $j$  and then behave as if nothing happened, so that the probability of neighbours of  $i$  not causally influenced by  $j$  observing each information set is the same, whether  $i$  follows  $\sigma_i^*$  or  $\sigma'_i$ . Therefore, those agents cannot punish  $i$  for the defection.

*Indistinguishable Evolving Graphs.* We say that an evolving graph  $G$  is indistinguishable from  $G'$  to agent  $i$  at round  $m$  if  $i$  acquires the same information about  $G$  and  $G'$ , regardless of the protocol followed by agents. Formally, given  $G \in \mathcal{G}^*$  and round  $m' \geq m$ , let  $\mathcal{G}_j^{m'}(G)$  be the set of round- $m'$  graphs that provide the same information to  $j$  about the round- $m'$  topology as  $G^{m'}$ , and let  $C_i^{m'}(G)$  be the set of agents  $j$  such that  $(j, m') \sim_i^G (i, m)$ ;  $G$  is indistinguishable from  $G'$  to  $i$  at  $m$  iff, for every round  $m' \leq m$ ,  $C_i^{m'}(G) = C_i^{m'}(G')$  and  $\mathcal{G}_j^{m'}(G) = \mathcal{G}_j^{m'}(G')$  for all  $j \in C_i^{m'}(G)$ .

## 4. WEAKEST ADVERSARY

We identify a necessary restriction on  $\mathcal{G}$  for enforcing accountability, and provide a protocol that enforces accountability assuming only this restriction and that pairwise exchanges are valuable.

### 4.1 Need for Timely Punishments

The weakest restriction on  $\mathcal{G}^*$  is called *timely punishment*.

In a nutshell, this property states that there is some known horizon  $\rho$  such that, if  $i$  defects some neighbour  $j$  at round  $m$ , then  $i$  may be punished before round  $m + \rho$ .

More precisely, for some bound  $\rho > 0$ , and for every  $G \in \mathcal{G}^*$ , agent  $i$ , and  $i$ -edge  $(j, m)$ , there must be a PO  $(l, m')$  of  $i$  in  $G$  for  $(j, m)$  such that  $m' < m + \rho$ . The need for this restriction is fairly intuitive. If the adversary is not restricted by timely punishments, then there is  $G \in \mathcal{G}^*$  and an agent  $i$  such that either (1) for some  $i$ -edge  $(j, m)$ , there is no PO of  $i$  in  $G$  for that  $i$ -edge, or (2) there is no limit on the time it takes between an interaction of  $i$  and a corresponding PO. In case (1),  $i$  can follow an evasive strategy to ensure that no agent capable of punishing  $i$  learns about the defection, thus never being punished. In case (2),  $i$  can delay a punishment for an arbitrarily long time; the problem here is the discount factor  $\delta$ : for an arbitrarily large constant  $d$ , there is an interaction of  $i$  such that, if  $i$  defects the neighbour and later follows an evasive strategy, then  $i$  is only punished after  $d$  rounds, and the utility loss of this punishment is discounted by  $\delta^d$ ; for a sufficiently large  $d$ , the immediate gain of defecting outweighs the loss. Unfortunately, some real networks do not always admit timely punishments. For instance, in a file sharing application, agents with similar interest may exchange files frequently, but occasionally they may interact with agents with different interests. If agents  $i$  and  $j$  have different interests and  $i$  happens to interact with  $j$ , then  $j$  may never be able to report a defection of  $i$  to agents with interests similar to  $i$ , which may be the only PO of  $i$ .

Theorem 1 shows that timely punishments are necessary to enforce accountability.

**THEOREM 1.** *If the adversary is not restricted by timely punishments, then there is no protocol that enforces accountability.*

**PROOF.** The proof is by contradiction. Suppose that  $\vec{\sigma}^*$  enforces accountability. Suppose also that the adversary is not restricted by timely punishments. For every  $\rho > 0$ , we can fix  $G \in \mathcal{G}^*$ , agent  $i$ , and  $i$ -edge  $(j, m)$  such that, for every PO  $(l, m')$  of  $i$  for  $(j, m)$  in  $G$ , we have  $m' - m \geq \rho$ . This implies that, for every round  $m' \geq m$  with  $m' < m + \rho$ , and every  $i$ -edge  $(l, m')$  with  $l \neq j$ ,  $(j, m) \sim_i^G (l, m')$  is false, since the existence of a PO  $(l, m')$  of  $i$  for  $(j, m)$  in  $G$  with  $m' < m + \rho$  is equivalent to the existence of one such  $(l, m')$  with  $(j, m) \sim_i^G (l, m')$ . Let  $\rho$  be such that  $y\delta^\rho/(1 - \delta) < 1$ , where  $y$  is the (bounded) maximum difference between utilities of a single interaction. Since  $\sigma_i^*$  enforces accountability,  $i$  is expected to cooperate at every round- $m$  information set  $I_i \in \mathcal{I}_i(G)$  consistent with  $\vec{\sigma}^*$  and  $G$ . Here,  $i$  may follow an evasive strategy  $\sigma'_i$  relative to  $\vec{\sigma}^*$ ,  $G$ , and  $(j, m)$  (see the full paper for details). Let  $\vec{\sigma}' = (\sigma'_i, \vec{\sigma}_{-i}^*)$ . By the properties of evasive strategies, for every round  $m' < m + \rho$  with  $m' \geq m$ , the round- $m'$  neighbours of  $i$  observe each round- $m' + 1$  information set with the same probability, whether  $i$  follows  $\sigma'_i$  or  $\sigma_i^*$ . Since these information sets specify the individual actions followed by and taken towards  $i$ , the expected utility of  $i$  in round  $m'$  is the same, whether agents follow  $\vec{\sigma}'$  or  $\vec{\sigma}^*$ , except  $i$  avoids at least the cost 1 of defecting  $j$  in round  $m$ . Moreover, the maximum utility difference in every round  $m' \geq m + \rho$  is  $yn$ . Therefore, we have

$$u_i(\vec{\sigma}^* | G, I_i) - u_i(\vec{\sigma}' | G, I_i) < -1 + \delta^\rho yn / (1 - \delta) < 0.$$

This contradicts the assumption that  $\vec{\sigma}^*$  enforces account-

ability, concluding the proof.  $\square$

Although the proof of this theorem is relatively straightforward, it is central for the remainder of the paper, as this result constrains the possible solutions that enforce accountability in dynamic networks.

## 4.2 A $\mathcal{G}^*$ -OAPE for Valuable Pairwise Exchanges with Timely Punishments

We now describe a protocol  $\vec{\sigma}^{\text{val}}$  that enforces accountability in a setting of valuable pairwise exchanges (defined below), assuming an adversary restricted by timely punishments. In valuable pairwise exchanges, we assume that agents can perform proportional punishments and punishment avoidance actions. This implies that agents can punish each other with a cost proportional to  $\pi$ , and they can avoid such punishments at the expense of not receiving the value from their neighbours. We also assume that the benefit-to-cost ratio of receiving/ sending a value is high. Specifically, we assume that  $\beta > 1 + \alpha + \rho\pi$  and  $\pi > n$ , where  $\rho$  is the constant in the definition of timely punishments. At the end of this section, we discuss exchanges that can be modelled in this way, assuming that agents exchange multiple messages per round and neglect download costs ( $\alpha = 0$ ).

We now describe  $\vec{\sigma}^{\text{val}}$ . In every round, agents exchange values and monitoring information that includes accusations revealing defections. These accusations are disseminated across the network, and used by agents to adjust the cost of a punishment applied to neighbours. More precisely, let  $\rho$  be as in the definition of timely punishments. At the beginning of round  $m$ , agent  $i$  keeps for every agent  $l$  and round  $m' \in \{m - \rho \dots m - 1\}$  a report indicating whether  $l$  defected some neighbour in round  $m'$ . We call a report indicating a defection an accusation. In a round- $m$  action, agents  $i$  and  $j$  exchange their reports and values, and apply punishments proportional to the number of accusations against each other. At the end of round  $m$ ,  $i$  updates its reports relative to every  $l \neq j$  and  $m' < m$  if  $j$  does not defect  $i$ , otherwise  $i$  emits an accusation against  $j$  for round  $m$ .

In the full paper, we prove Theorem 2, which shows that  $\vec{\sigma}^{\text{val}}$  enforces accountability in valuable pairwise exchanges, under two minimal assumptions: (1) the adversary is restricted by timely punishments, which as we have seen is strictly necessary, and (2) agents are *sufficiently patient*, i.e., the factor  $\delta$  is sufficiently close to 1, which is a standard assumption in proofs of Folk Theorems, and is necessary for future losses of punishments to always outweigh the gains of deviating in the present (recall that future losses are discounted to the present by  $\delta$ ). The proof shows that a defection of  $i$  is always matched by a punishment that occurs after at most  $\rho$  rounds.  $i$  gains at most  $n$  by defecting neighbours in a round, while losing at least  $\delta^\rho \pi$ . Given that  $\beta > 1 + \alpha + \rho\pi$  and  $\pi > n$ , if  $\delta$  is sufficiently close to 1, then the loss outweighs the gain. Moreover, if  $i$  avoids the cost of a punishment and of sending and receiving messages (at most  $1 + \alpha + \rho\pi$ ),  $i$  does not receive the value, thus losing  $\beta$ . Since  $i$  can never influence the reports relative to itself,  $i$  never gains by deviating.

**THEOREM 2.** *If the adversary is restricted by timely punishments and agents are sufficiently patient, then  $\vec{\sigma}^{\text{val}}$  enforces accountability in valuable pairwise exchanges.*

## 4.3 Examples of Valuable Pairwise Exchanges

We now provide examples of interactions that meet the restrictions of valuable pairwise exchanges. First, we consider an interaction without cryptography. Then, we show how cryptography can decrease the number of restrictions on the utility.

### Without cryptography.

Consider that every neighbouring agents  $i$  and  $j$  can exchange messages in three phases per round and neglect download costs, which may be the case if bandwidth is asymmetric. In phase 1,  $i$  sends reports indicating the number  $c_j$  of accusations against  $j$ , and similarly  $j$  sends the number  $c_i$  of accusations against  $i$ . In phase 2,  $i$  and  $j$  send  $c_i$  and  $c_j$  penance messages that cost  $\pi$  each, respectively. In phase 3, they exchange their values only if both agents have sent all required information in previous phases. In this setting, agents can adjust the size of penances such that  $\pi > n$ . Suppose that the communication costs are normalized such that the total cost of sending phase 1, 2, and 3 messages is 1. If  $i$  sends all requested information in the first two phases,  $i$  does not avoid the punishment of sending the  $c_i$  penance messages, while avoiding at most the cost 1 of defecting  $j$  by not sending the value in phase 3. As shown by the proof of Theorem 2,  $i$  does not gain from defecting.  $i$  can avoid the punishment by not sending the requested messages in the first two phases, but in this case  $i$  does not receive the value from  $j$  in phase 3, also not gaining from this deviation.

### With cryptography.

The previous interactions require knowledge of  $\beta$  in order to appropriately define the size of a penance message, and require  $\beta = \Omega(n\rho)$ , making it restrictive in practice if  $n$  is large. We can mitigate these problems using the technique of delaying gratification from [18]. Instead of exchanging values in phase 3 of round  $r$ , neighbours  $i$  and  $j$  exchange the values ciphered with private keys in phase 1 along with monitoring information. In phase 2, they still send the penances and in phase 3 they exchange the keys. As before, if messages are omitted in phase 1 or 2, then the agents send no further messages. Let  $\alpha^\kappa$  be the cost of sending the key. This mechanism only requires  $\pi > \alpha^\kappa$  and  $\beta > 1 + n\rho\pi$ . Thus, it suffices that  $\beta > 1 + \epsilon$ , where  $\epsilon = np\alpha^\kappa$ . If  $\alpha^\kappa \ll 1$ , then  $1 + \epsilon$  is close to the optimal restriction of  $\beta > 1$ . It is also possible to adjust  $\pi$  without knowing  $\beta$ : we can define the size of a penance to be larger than that of a key, but smaller than the value. The arguments that show that this mechanism is a  $\mathcal{G}^*$ -OAPE are the same as in the proof of Theorem 2.

## 5. GENERAL PAIRWISE EXCHANGES

We now address the problem of enforcing accountability without making the assumption that pairwise exchanges are valuable. We consider the least restrictive assumptions about utility and individual actions available to agents. Namely, we consider the smallest benefit-to-cost ratio of receiving/ sending values. We still need the benefit  $\beta$  to be larger than the total costs  $1 + \alpha$  of sending and receiving messages, or else agents would not have incentives to engage in exchanges. Second, we do not assume that agents neglect download costs ( $\alpha \geq 0$ ), nor that they communicate in multiple steps. Therefore, proportional punishments and

punishment avoidance are not available actions. We still assume that agents can defect, cooperate, or (actively) punish other agents. Since there is a trivial one-shot implementation of active punishments, where an agent sends garbage of the size of the value instead of the value, we consider that a punishment causes a utility loss of  $\pi \geq \beta$  to the punished agent, whereas the punisher agent incurs a cost of 1.

With this in mind, we say that a protocol enforces accountability in general pairwise exchanges if it is an equilibrium for all utilities such that  $\beta > 1 + \alpha$ , and only has agents following the three aforementioned actions. We show in the full paper that non-safe protocols cannot in general enforce accountability. In this section, we identify a necessary and a sufficient condition for enforcing accountability with safe-bounded protocols in general pairwise exchanges. The results are generalized for non-bounded protocols in the full paper.

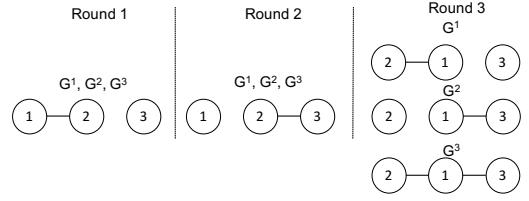
## 5.1 Need for Eventual Distinguishability

We identify a necessary restriction on  $\mathcal{G}^*$  to enforce accountability with safe-bounded protocols in general pairwise exchanges, called eventual distinguishability. Due to lack of space, we only provide an informal definition of this condition and an intuition for its necessity, deferring the formal definition, proof, and generalization for non-bounded protocols to the full paper.

Intuitively, for every interaction between an agent  $i$  and a neighbour  $j$ , if  $i$  defects  $j$ , then  $i$  must be punished by at least one agent in a future interaction, or else  $i$  gains by defecting. In this section, we show that there is a problem if  $i$  is punished by two or more agents for every defection. Namely, if  $i$  repeatedly defects neighbours and is punished by two or more agents for each of those defections, then the number of future punishments that  $i$  expects to receive in the future grows without bound as  $i$  keeps defecting neighbours. Consequently, the protocol cannot be bounded, since there is no bound on the time it takes for punishments to end after  $i$  stops deviating. The condition of eventual distinguishability establishes restrictions on  $\mathcal{G}^*$  that allow us to devise  $\mathcal{G}^*$ -OAPE protocols where an agent is punished only once for each defection.

More precisely, we say that the *adversary is restricted by eventual distinguishability* if for every evolving graph  $G$  in  $\mathcal{G}^*$  and agent  $i$ , eventually  $i$  stops having interactions in  $G$  where  $i$  defects some neighbour  $j$ , and then expects to be punished by two or more agents for that defection, because each of those agents cannot distinguish between two evolving graphs where it should and should not punish  $i$ . Specifically, the formal definition of eventual distinguishability has two parts: (1) a definition of *indistinguishable rounds*, in which a defection of some agent is matched by more than one future punishment, and (2) the requirement that indistinguishable rounds must eventually stop occurring.

Fig. 2 depicts an example of an indistinguishable round. There are three agents numbered 1 to 3. The adversary may generate three alternative evolving graphs  $G^1$ ,  $G^2$ , and  $G^3$ . In round 1, agent 1 interacts with agent 2. In round 2, agent 2 interacts with agents 3. In round 3, the interactions depend on the evolving graph: ( $G^1$ )  $i$  interacts only with 2, ( $G^2$ )  $i$  interacts only with 3, and ( $G^3$ )  $i$  interacts with both 2 and 3. If the protocol is safe and enforces accountability in general pairwise exchanges, then 1 must send a message to 2 in round 1. If 1 defects 2, then 1 must be punished



**Figure 2: Agent 2 cannot distinguish  $G^1$  from  $G^3$  at round 3, and agent 3 cannot distinguish  $G^2$  from  $G^3$  at round 3.**

at least once. This is because, by defecting, 1 avoids the cost 1 of sending a message, so this gain must be matched by a future loss of at least  $\pi > 1$ . Suppose that round 3 is the only opportunity to punish  $i$  for defecting 2 in round 1, in time. Then, agent 1 must be punished by agent 2 in  $G^1$  and by agent 3 in  $G^2$ , respectively. In the latter case, since agent 3 does not directly observe the defection of 1, agent 2 must tell agent 3 in round 2 about the defection of agent 1 when the evolving graph is  $G^2$ . Since agent 2 cannot distinguish  $G^2$  from  $G^3$  at round 2, it must also tell agent 3 about the defection of 1 in  $G^3$ . If the only information about the topology is the identity of the neighbours, then neither agent 2 nor agent 3 can distinguish  $G^1$  from  $G^3$  nor  $G^2$  from  $G^3$ , respectively. This means that in  $G^3$  both agents 2 and 3 must punish 1 for defecting 2 in round 1. Therefore, after round 1, agent 1 expects to be punished by two neighbours, even though 1 only defects one neighbour.

The above example shows that there is a round-2 information set  $I_1 \in \mathcal{I}_1(G^3)$  such that, according to the information in  $I_1$  available to 1, the expected number of punishments of 1 conditional on  $G^3$  and  $I_1$  is 2. If this type of interactions keeps occurring in  $G^3$ , then, for an arbitrarily large number  $c$ , there is an information set  $I_1 \in \mathcal{I}_1(G^3)$  such that, basing on the information in  $I_1$ , agent 1 expects to be punished by at least  $c$  neighbours after  $I_1$ . The fact that only agent 1 deviates plays a key role here: by the definition of equilibrium, we only need to ensure that 1 never gains by defecting a neighbour at round 1 provided that other agents do not deviate afterwards. If multiple agents deviate after round 1 (e.g., agent 2 defects 3 in round 2), then 1's defection in round 1 can be forgiven, since such behaviour is unexpected at the time 1 decides to defect 2 in round 1. The assumption that protocols are safe is also crucial, since non-safe protocols are not susceptible to the aforementioned problem. Unfortunately, as we show in the full paper, we cannot in general devise a non-safe protocol that enforces accountability.

In the full paper, we prove Theorem 3. The proof shows that, if the adversary is not restricted by eventual distinguishability, then, for some agent  $i$  and evolving graph  $G$  in  $\mathcal{G}^*$ , if  $i$  always defects all neighbours in  $G$ , then, in an indistinguishable round, the expected number of future punishments after the defections is strictly larger than before, whereas in all remaining rounds the expected number of punishments never decreases. This is sufficient to show that the expected number of punishments is unbounded, and hence that no safe protocol that enforces accountability in general pairwise exchanges can be bounded.

**THEOREM 3.** *If the adversary is not restricted by eventual distinguishability, then there is no safe-bounded protocol that*

enforces accountability in general pairwise exchanges.

## 5.2 A $\mathcal{G}^*$ -OAPE for General Pairwise Exchanges with Connectivity

We now describe a restriction of connectivity on  $\mathcal{G}^*$  that ensures that the adversary is restricted by eventual distinguishability. We also show that under this restriction there is a safe-bounded protocol that enforces accountability in general pairwise exchanges. To understand the restriction, it is useful to first recall the scenario of Figure 2. In this scenario, the problem arises because neither agent 2 can distinguish  $G^1$  from  $G^3$  at round 3 nor agent 3 can distinguish  $G^2$  from  $G^3$  at round 3, and thus they cannot coordinate their actions to punish 1 only once. This problem can be avoided if agents can learn the degree of their neighbours prior to deciding whether to punish them. However, the knowledge of the degree is not sufficient to satisfy eventual distinguishability, since for instance we may have a scenario similar to the one depicted in Figure 2, except agents 2 and 3 do not punish 1 in the same round. This can be avoided if it is always the case that agents 2 and 3 always punish 1 in the same round. The problem also does not arise if, for instance, agent 2 is the first to punish 1 and then warns agent 3 of this fact before 3 interacts with 1.

More generally, it suffices that, for every  $G \in \mathcal{G}^*$ , agent  $i$ , and round  $m$ , there is an horizon  $\rho$  such that the round- $m$  neighbours of  $i$  causally influence every round- $m + \rho$  neighbour of  $i$  between  $m$  and  $m + \rho$  without interference from  $i$ . This condition is exactly met with  $\rho = n$  when  $\mathcal{G}^*$  is restricted by a condition similar to 1-connectivity from [15]: we say that the adversary is *restricted by connectivity* iff agents know the degree of their neighbours and, for every  $G \in \mathcal{G}^*$ , agent  $i$ , and round  $m$ , the graph obtained from  $G^m$  by removing the edges to  $i$  is connected. This condition is also met by overlays for gossip dissemination such as [18, 17, 9].

We now define a safe-bounded protocol  $\vec{\sigma}_i^{\text{gen}}$  that enforces accountability in general pairwise exchanges, assuming that the adversary is restricted by connectivity. Fix  $G \in \mathcal{G}^*$  and let  $\deg_i^m$  denote the degree of  $i$  in  $G^m$ . At every round  $m$ , neighbouring agents always exchange monitoring information, and they follow punishment individual actions with a probability proportional to past deviations. Monitoring information includes reports and numbers of pending punishments. Specifically, for each round  $m' < m$  and pair of agents  $(j, l)$ , a report relative to  $m'$  and  $(j, l)$  specifies whether  $j$  interacted with  $l$  in round  $m'$ , and whether  $l$  defected  $j$ . For each  $c \in \{1 \dots n\}$  and agent  $j$ , agents keep the number of pending punishments to be applied to  $j$  in periodic rounds  $(c + kn)_{k \geq 0}$ . Before interacting with  $j$  in round  $m$ ,  $i$  determines whether  $j$  should be punished. For this,  $i$  updates the number  $x$  of pending punishments for the period that includes  $m$ . If  $m \leq n$ , then  $x = 0$ . Otherwise, let  $x'$  be the previous number of pending punishments resulting from an identical update prior to round  $m - n$ . Given the round- $m - n$  reports,  $i$  determines  $\deg_j^{m-n}$ . Then,  $i$  sets  $x$  to  $\max(0, x' - \deg_j^{m-n})$ , and adds  $\deg_j^{m-n}$  iff  $j$  defected some neighbour in  $m - n$ .  $i$  punishes  $j$  with probability  $\min(1, x/\deg_j^m)$ . (This is where the knowledge of degree comes into play.) After the interaction,  $i$  emits a report indicating that the interaction occurred in round  $m$  and signalling whether  $j$  defected  $i$ .  $i$  also updates its monitoring information basing on the information sent by its neighbours.

---

### Algorithm 1 $\vec{\sigma}_i^{\text{gen}}$

---

```

1: for all  $j, m$  do
2:    $\text{AP}_j^m \leftarrow 0$ 
3:   for all  $l$  do
4:      $\text{RP}_{jl}^m \leftarrow \perp$ 

5: Upon round  $m$ 
6:   for all neighbour  $j$  do
7:      $pr \leftarrow \min(1, \text{AP}_j^m / \deg_j^m)$ 
8:     With probability  $pr$ , cooperate            $\triangleright$  Punishment
9:     Otherwise, punish                       $\triangleright$  Cooperation
10:  End

11: After round  $m$ 
12:   for all neighbour  $j$  do
13:     if  $j$  defects  $i$  in  $m$  then
14:        $\text{RP}_{ij}^m \leftarrow \text{Bad}$ 
15:     else
16:        $\text{RP}_{ij}^m \leftarrow \text{Good}$ 
17:       for all  $m' \in \{m - n + 1 \dots m - 1\}$  do
18:          $\text{AP}_j^{m'} \leftarrow \text{maximum between } \text{AP}_j^{m'} \text{ and values sent}$ 
           by each  $l \neq i$  do
19:           Cap  $\text{AP}_j^{m'}$  to lie in  $\{0 \dots n - 1\}$ 
20:           for all  $l \neq i$  do
21:             if  $\text{RP}_{lj}^{m'} = \perp$  and some  $k \neq j$  sent  $\text{RP}_{lj}^{m'}|_k \neq \perp$ 
               then
22:                $\text{RP}_{lj}^{m'} \leftarrow v \neq \perp$  deterministically selected
               among received values
23:             for all agent  $j \neq i$  do
24:               if  $m \geq n$  then
25:                  $\text{deg} \leftarrow \#\{l \neq j \mid \text{RP}_{lj}^{m-n+1} \neq \perp\}$   $\triangleright$  Degree  $\deg_j^{m-n+1}$ 
26:                  $\text{AP}_j^{m+1} \leftarrow \max(0, \text{AP}_j^{m-n+1} - \text{deg})$ 
27:                 if exists  $l \neq j$  such that  $\text{RP}_{lj}^{m-n+1} = \text{Bad}$  then
28:                    $\text{AP}_j^{m+1} \leftarrow \text{AP}_j^{m+1} + \text{deg}$ 
29:  End

```

---

For each report relative to round  $m' < m$  not older than  $m - n + 1$  and pair  $(k, l)$ , if  $i$  does not have a report relative to  $m'$  and  $(k, l)$  and receives a new report from  $j \neq l$ , then  $i$  stores this report. In addition, for each period  $c$  not including  $m$ ,  $i$  updates the number of pending punishments relative to every  $l \neq i$  and  $c$  to the maximum between its value and the value sent by every neighbour  $j \neq l$ , capping it to lie in  $\{0 \dots n - 1\}$ .

We present in Alg. 1 the pseudo-code for the strategy  $\vec{\sigma}_i^{\text{gen}}$  of agent  $i$ . We use two variables: AP and RP. For each round  $m$ ,  $i$  keeps a number  $\text{AP}_j^m$  of pending punishments to be applied to  $j$  in rounds  $m, m + n, m + 2n, \dots$ , initially equal to 0 and never larger than  $n - 1$ . Also,  $i$  keeps a report  $\text{RP}_{jl}^m \in \{\text{Good}, \text{Bad}, \perp\}$  per pair of nodes  $(j, l)$  and round  $r$  signalling whether in round  $m$ : (i)  $j$  did not interact with  $l$  ( $\perp$ ); (ii)  $l$  interacted and defected  $j$  ( $\text{Bad}$ ); or (iii)  $l$  interacted and did not defect  $j$  ( $\text{Good}$ ).  $\text{RP}_{jl}^m$  is initialized to  $\perp$ . Notice that these variables can be implemented by a finite state machine: at each round  $m$ , agents only need to store and forward information relative to each tuple  $(i, j, c)$  for  $c \in \{0 \dots n - 1\}$ , corresponding to  $\text{RP}_{ij}^{m-c}$ , and information relative to each pair  $(i, c)$  for  $c \in \{0 \dots n - 1\}$ , corresponding to  $\text{AP}_i^{m-c}$ . Therefore,  $\vec{\sigma}_i^{\text{gen}}$  is a safe-bounded protocol. For the sake of exposition, we opt to not represent the state in this compact form.

This definition has the following properties when agents follow  $\vec{\sigma}_i^{\text{gen}}$  at every round- $m$  history  $h$ : (1)  $i$  cannot influence monitoring information that determines punishments

applied to  $i$ , (2) we match each defection of  $i$  in round  $m$  to at least one punishment in future rounds, (3) a defection in round  $m$  triggers additional punishments to be applied in rounds  $m+n, m+2n \dots$ , and (4) the delay of additional punishments is bounded by  $O(n^2)$ . (1) ensures that  $i$  does not gain from lying about monitoring information. (2) ensures that, even if  $i$  saves the cost 1 of sending messages,  $i$  loses at least  $\beta > 1$ . (3) and (4) guarantee that this loss is discounted to the present by a lower bounded factor  $\delta^{n^2}$ . This implies that, if agents are sufficiently patient (i.e.,  $\delta$  is sufficiently close to 1), then  $i$  prefers not to defect. Therefore, Theorem 4 holds. (See the full paper for a proof.)

**THEOREM 4.** *If the adversary is restricted by connectivity and agents are sufficiently patient, then  $\vec{\sigma}^{\text{gen}}$  enforces accountability in general pairwise exchanges.*

### 5.3 Avoiding Prior Knowledge of Degree

The knowledge of degree can be a restrictive assumption in practice. We now discuss one type of interactions in general pairwise exchanges, where agents only have to know the identity of their neighbours in order for  $\vec{\sigma}^{\text{gen}}$  to be an equilibrium. We only assume that agents engage in multiple message exchanges per interaction. The idea is for neighbouring agents to reveal their degrees before sending the values, and then punish each other proportionally to the degree.

We need at least two communication phases, so that neighbours  $i$  and  $j$  may first reveal their degree in phase 1, and then exchange values in phase 2 and punish each other accordingly. Unfortunately, two phases is not enough because  $i$  may lie about the degree. In particular, in phase 1,  $i$  may declare a higher degree than the real one to decrease the probability of being punished by  $j$ . We can address this by including in the report information about the declared degrees, and then punish agents that lie about their degrees. This is still not sufficient though, because of the following scenario. Suppose that  $i$  only has one neighbour  $j$  and only one pending punishment. If  $i$  does not lie, then  $j$  punishes  $i$  with probability 1 due to having a single pending punishment. If  $i$  declares a degree of  $n - 1$  instead, then the probability of being punished by  $j$  is only  $1/(n - 1)$ . This yields an increase in the expected benefits from 0 to  $\beta(n - 2)/(n - 1)$ . In addition, suppose that  $i$  defects  $j$  by omitting messages in phase 2. The expected future loss must be at most  $\beta$ , to avoid the problem identified in the proof of the need for eventual distinguishability. Since we only assume that  $\beta > 1 + \alpha$ , the loss may be lower than the gain.

The problem is that  $i$  sends the degree before incurring the cost of sending messages in phase 2. We can avoid this by using the same technique of Section 4.3. We need three communication phases. In phase 1, agents exchange monitoring information and the values ciphered with random private keys. In phase 2, they reveal the degrees. Finally, in phase 3, they decide whether to cooperate by sending the private keys, or to punish by sending arbitrary keys. Let  $\alpha^\kappa$  be the cost of sending a key, such that the cost of sending phase 1 and 2 messages plus  $\alpha^\kappa$  is 1. Agent  $i$  is punished for defecting  $j$  in phases 1 or 2 by being punished in a later stage; if  $i$  is sufficiently patient, then as in the proof of Theorem 4  $i$  does not gain by deviating in the first two phases. The maximum utility gain obtained by  $i$  is when  $i$  has only one neighbour  $j$ ,  $i$  lies to  $j$  by saying that its degree is  $n - 1$ , and then defect  $j$  in phase 3. The maximum gain is  $\beta(n - 2)/(n - 1) + \alpha^\kappa$ ,

whereas the future loss is at least  $\delta^{n^2}\beta$ . If  $\alpha^\kappa < \beta/(n - 1)$ , then the gain is less than  $\beta$ , and the future loss outweighs the gain for a  $\delta$  sufficiently close to 1. Therefore, we have a protocol that enforces accountability.

### 5.4 Complexity

The bit complexity of  $\vec{\sigma}^{\text{gen}}$  is  $O(n^3)$ : each message carries  $n^3$  reports, with one report per pair of agents and round in  $m - n \dots m - 1$ , carries  $n^2$  accusations, and carries  $n^2$  numbers of pending punishments. The maximum delay of punishments is  $O(n^2)$ . We can improve the complexity by assuming further restrictions on  $\mathcal{G}^*$  and on the computational ability of agents. Specifically, the factors  $n^3$  and  $n^2$  are a function of (1) the maximum delay  $n$  of disseminating information relative to an agent, and (2) the maximum number  $n$  of agents relative to which an agent has to forward information. (1) can be improved by considering more restrictive assumptions about  $\mathcal{G}^*$ . As discussed in the definition of adversary restricted by connectivity, we only need that the neighbours of any given agent  $i$  in round  $m$  can causally influence all the neighbours of  $i$  in round  $m + \rho$  for some constant  $\rho$ . If the evolving graphs satisfy locality properties that ensure that the neighbours of agent  $i$ 's neighbours are likely to be  $i$ 's neighbours in the near future, then  $\rho$  can be significantly smaller than  $n$ . Then, the bit complexity becomes  $O(\rho n^2)$  and the maximum delay of punishments becomes  $O(\rho n)$ . Small-world networks such as social networks, which have a high clustering coefficient, are well known examples that satisfy such locality properties [11, 25].

Interestingly, locality properties also allow us the improve on (2). Agents only have to forward information relative to an agent  $i$  and round  $m$  in the  $\rho$  rounds following  $m$ . This is necessary to ensure that critical information reaches the agents capable of carrying a punishment. If during every period of  $\rho$  rounds each agent were only causally influenced by a limited number  $c$  of agents, then agents only needed to forward information relative to  $c$  agents, and the bit complexity and delay of punishments would be  $O(\rho c^2)$  and  $O(\rho c)$ , respectively. Unfortunately, such reduction on the amount of information each agent forwards introduces a congestion problem. Specifically, we cannot let agents sending messages of varying size, since that would incentivize them to always forward the minimum required information to save bandwidth. Therefore, they must forward messages of fixed size, which implies that agents may have to discard information if they cannot fit all received information in a single message. This gives the opportunity for an agent  $i$  to deliberately generate false reports (or accusations) that flood the network, causing other agents to discard accusations against  $i$ . We can address this issue by appending to each report and accusation a signature of the issuer. A report of an interaction between  $i$  and  $j$  has issuer  $i$ , so a valid report of this interaction must contain a signature of  $i$ . The same applies to accusations and numbers of pending punishments.

## 6. DISCUSSION

Our results provide technical insight on the type of dynamic networks that can support pairwise exchanges in equilibria strategies. The need for timely punishments means that accountability cannot be enforced in some dynamic networks such as certain overlays for file-sharing. If connec-

tivity is ensured, accountability may be enforced, even for general exchanges. Although for this scenario we have assumed knowledge of degree prior to interactions, this can be relaxed if agents can exchange multiple messages per round. Finally, if exchanges are valuable, timely punishments are enough to enforce accountability, which opens the door to protocols that enforce accountability in a wide variety of dynamic networks. There are multiple open questions to be addressed in future work. It would be interesting to prove a stronger condition that would close the gap between an adversary restricted by distinguishability and one restricted by connectivity. Another open issue is collusion. We believe that both the necessary and sufficient conditions presented in this paper could be strengthened by generalizing the notion of causal influence without interference from individual agents to the absence of interference from members of a coalition. Given these conditions, the  $\mathcal{G}^*$ -OAPE strategies are resilient to collusion.

**Acknowledgements:** This work has been partially supported by Fundação para a Ciência e Tecnologia (FCT) through projects with references PTDC/EEI-SCR/1741/2014 (Abyss) and UID/CEC/50021/2013.

## 7. REFERENCES

- [1] I. Abraham, D. Dolev, R. Gonen, and J. Halpern. Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. PODC'06, pages 53–62. ACM, 2006.
- [2] I. Abraham, D. Dolev, and J. Halpern. Distributed protocols for leader election: A game-theoretic perspective. DISC'13, pages 61–75. Springer-Verlag, 2013.
- [3] Y. Afek, Y. Ginzberg, S. Landau Feibis, and M. Sulamy. Distributed computing building blocks for rational agents. PODC'14, pages 406–415. ACM, 2014.
- [4] B. Cohen. Incentives build robustness in bittorrent. P2PEcon'03, Berkeley, CA, USA, June 2003.
- [5] A. Fabrikant, A. Luthra, E. Maneva, C. H. Papadimitriou, and S. Shenker. On a network creation game. PODC'03, pages 347–351. ACM, 2003.
- [6] I. Fainmesser. Community structure and market outcomes: A repeated games in networks approach. *American Economic Journal: Microeconomics*, 4(1):32–69, 2012.
- [7] I. Fainmesser and D. Goldberg. Cooperation in partly observable networked markets repeated games played in a network. UFAE and IAE working papers, Brown University, 2012.
- [8] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for peer-to-peer networks. EC'04, pages 102–111, New York, NY, USA, 2004. ACM.
- [9] R. Guerraoui, K. Huguenin, A. Kermarrec, M. Monod, and S. Prusty. Lifting: lightweight freerider-tracking in gossip. Middleware'10, pages 313–333. ACM, 2010.
- [10] J. Halpern and V. Teague. Rational secret sharing and multiparty computation: Extended abstract. STOC'04, pages 623–632. ACM, 2004.
- [11] P. W. Holland and S. Leinhardt. Transitivity in structural models of small groups. *Small Group Research*, 2(2):107–124, 1971.
- [12] S. Jun and M. Ahamad. Incentives in bittorrent induce free riding. P2PECON'05, pages 116–121. ACM, 2005.
- [13] S. Kamvar, M. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th International Conference on World Wide Web*, pages 640–651, Budapest, Hungary, 2003.
- [14] D. Kreps and R. Wilson. Sequential equilibria. *Econometrica*, 50(4):863–894, 1982.
- [15] F. Kuhn, N. Lynch, and R. Oshman. Distributed computation in dynamic networks. STOC'10, pages 513–522. ACM, 2010.
- [16] D. Levin, K. LaCurts, N. Spring, and B. Bhattacharjee. Bittorrent is an auction: analyzing and improving bittorrent's incentives. In *Proceedings of the ACM SIGCOMM 2008 conference on Applications, technologies, architectures, and protocols for computer communication*, SIGCOMM'08. ACM, Aug. 2008.
- [17] H. C. Li, A. Clement, M. Marchetti, M. Kapritsos, L. Robison, L. Alvisi, and M. Dahlin. Flightpath: obedience vs. choice in cooperative services. OSDI'08, pages 355–368. Usenix Association, 2008.
- [18] H. C. Li, A. Clement, E. L. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin. BAR gossip. OSDI'06, pages 191–204. Usenix Association, 2006.
- [19] G. Mailath and L. Samuelson. *Repeated Games and Reputations*. Oxford University Press, 2007.
- [20] T. Moscibroda, S. Schmid, and R. Wattenhofer. On the topologies formed by selfish peers. PODC'06, pages 133–142. ACM, 2006.
- [21] G. Neglia, G. Presti, H. Zhang, and D. Towsley. A network formation game approach to study bittorrent tit-for-tat. NET-COOP'07, pages 13–22. Springer-Verlag, 2007.
- [22] M. Osborne and A. Rubinstein. *A course in game theory*. The MIT Press, 1994.
- [23] X. Vilaca and L. Rodrigues. Accountability in dynamic networks. Arxiv preprint arXiv:1602.03885, May 2016.
- [24] X. Vilaca and L. Rodrigues. On the range of equilibria utilities of a repeated epidemic dissemination game with a mediator. ICDCN '15, pages 19:1–19:10. ACM, 2015.
- [25] D. J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442, 1998.
- [26] E. Wong and L. Alvisi. What's a little collusion between friends? PODC'13, pages 240–249. ACM, 2013.

# Secure Multiparty Construction of a Distributed Social Network

Varsha Bhat Kukkala  
varsha.bhat@iitrpr.ac.in

Jaspal Singh Saini  
jaspal.singh@iitrpr.ac.in

S.R.S. Iyengar  
sudarshan@iitrpr.ac.in

Department of Computer Science and Engineering  
Indian Institute of Technology Ropar  
Punjab, India - 140001

## ABSTRACT

The advancement in technology has resulted in a better connected society. These connections foster social interactions that result in an emergent structure. This structure is popularly termed as a social network and is an integral component of study, in the field of network science. However, the study of these social networks is limited to the availability of data on the underlying social interactions. Privacy concerns restrict the access to network data with sensitive information. Networks that capture the relations such as trust, enmity, sexual contact, are a few examples of sensitive networks. A study of these sensitive networks is important in unraveling the behavioral aspects of the concerned individuals. The current paper proposes a multiparty computation algorithm that allows the construction of an unlabeled random isomorphic version of a distributedly held network. The protocol is proven to be secure in the presence of the extended arithmetic black-box, which supports the operations of addition, multiplication, comparison and equality checks.

## CCS Concepts

- Security and privacy → Cryptography;

## Keywords

multiparty computation; social networks; distributed algorithms

## 1. INTRODUCTION

The concept of a social network is no more limited to the field of sociology, where it was first introduced. Today, the idea is widely applied to a myriad of domains, such as biology, chemistry, marketing, economics and epidemiology. The presence of common topological characteristics, across different networks, is the reason for its wide applicability. Social networks are modeled as graphs with social entities represented as nodes and the edges of the graph cap-

turing the interrelationship between the entities. Some of the most frequently studied networks include friendship networks, human-contact networks, communication networks, citation networks, etc. Studying these networks has helped better understand the phenomena of information cascades, communication patterns, spread of diseases and influence. Several online social networks, like Facebook, Twitter and LiveJournal, have constantly been a playground for analyzing the network structure and its implications. However, a social network innately houses sensitive information of the concerned individuals. The resulting privacy concerns have been a major impediment to the study of many networks.

A study of sensitive interrelationships like trust, hatred, sexual contact, etc., can have an unforeseen impact on our understanding of the behavioral patterns observed across individuals. For example, the amount of hatred fostered in a team can have correlations to the team's overall productivity. However, gathering data of the hate network would be a challenge, as individuals would not be willing to reveal the sensitive information about the team members they dislike. A surveillance of the trust network, over time, in an organization can be used for building better teams and selecting team leaders. Most commonly, studies conducted on networks with sensitive information acquire the data through surveys [12], which is further anonymized to ensure privacy. The fear of sensitive information being leaked prevents most of the users from sharing their private information [10] or could even lead to reporting false information. To address these drawbacks of surveys, there is a necessity for constructing a protocol that can generate the underlying network while guaranteeing the privacy of the participating individuals.

Constructing the underlying network securely would require amalgamating the data that is available distributedly. It must be done in a way that privacy and integrity of the inputs is ensured, while guaranteeing the correctness of the generated output. This is precisely what constitutes a multiparty computation (MPC) protocol. It involves a set of parties, who follow a protocol (specific sequence of instructions) for computing a function of their private data. The process must ensure that nothing but the final result is revealed. The first MPC protocol was proposed by Yao [16], which allowed two parties to compare and determine who among the two is richer, without revealing each other's wealth. Multiparty computation has evolved as a separate branch of cryptography, whose tools and techniques have been used in addressing numerous problems, such as, computing approximations on distributed data, auctions, private matching and set in-

ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of a national government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only.

*ICDCN '17, January 04-07, 2017, Hyderabad, India*

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007783>

tersection, secure rank computation, privacy preserving data classification and data mining.

The contribution of the current work is to provide an isomorphic graph construction protocol that is secure in accordance to the requirements of any standard MPC protocol. The proposed solution assumes an extended arithmetic black-box functionality, various implementations of which exist in literature. The protocol allows the construction of the unlabeled version of the weighted directed graph on  $n$  individuals, who are participating in the protocol. Each individual, henceforth referred to as a party, reports her adjacency list (all her out-going edges) as her private input. The algorithm guarantees that the parties do not learn any additional information apart from the data that can be gathered from just their input and output. The proposed protocol can be easily modified to construct an unweighted and/or undirected graph as well.

## 2. RELATED WORK

Securely computing algorithms on a network has been studied in the recent past. Brickell and Shmatikov [5] constructed two party protocols for computing all pair shortest paths and single source shortest paths securely, in the semi-honest adversarial model. In this protocol, each party possesses a graph, such that both the parties are interested in computing algorithms over the union of the two input graphs. Hu, Chow and Lau [13] discuss on how one can detect people belonging to the same community with minimum information being leaked. Such a detection allows to suggest friends in a social network. Zeng et al. [17] also propose a technique for secure link prediction in online social networks. Aly et al. [1] study the problem of computing shortest paths in a graph securely. Aly and Vyve [2] address the problem of finding minimum mean cycle and the minimum cost flow problems in a multiparty setting. Blanton et al. [4] propose a data oblivious method for computing graph algorithms, such as BFS, shortest paths, minimum spanning tree and network flow problems. All these works target on securely performing specific graph algorithms, while keeping the underlying network hidden. On the contrary, the current work focuses on securely releasing the underlying network itself. This is more applicable when the target analysis is not predetermined.

Securely generating the underlying graph has been previously studied by Frikken and Golle [9]. It is assumed that the network information is held in a distributed manner, where each individual possesses some partial information of the network. The drawbacks of the protocol is that it uses special parties called authorities, who help compile the collected data into the required graph. Also, the use of threshold Elgamal encryption scheme and re-encryption mix nets in the protocol, amounts to increased communication and computation cost. It is to be noted that their protocol is restricted to the cryptographic security model. The protocol proposed in the current paper avoids the use of dummy parties and is secure in the  $\mathcal{F}_{ABB}$  hybrid model, thereby overcoming the above mentioned drawbacks. Bhat et al. [14] propose an information theoretic solution for compiling an isomorphic version of a distributedly held graph in the semi-honest setting with a threshold of  $\sqrt{n}$  corrupt parties. The protocol proposed in the current work can withstand a better threshold of corrupt parties, which depends on the implementation of the arithmetic black-box assumed.

## 3. PRELIMINARIES

In this section we introduce the notations and definitions used throughout the paper. We begin by defining a multi-party computation protocol as an algorithm, using which a set of  $n$  parties  $P_1, P_2, \dots, P_n$  can *securely* compute a function  $f$  over their private information. We will consider the field  $\mathbb{F}_p$  (or any other finite field will also suffice) with the modular operations of addition (+) and multiplication (\*) for arithmetic henceforth. A set of parties are said to be *corrupt* if they collaborate to reveal information about the set of honest parties i.e. parties that are not corrupt. In order to model corruption, we assume the presence of a central adversary, who controls all corrupt parties. A protocol is said to be secure in the malicious adversarial model if it is *correct*, *private* and *robust*. A detailed discussion on security of MPC protocols is available in [7], however we briefly describe the security requirements below:

- *Correctness* of a protocol guarantees that the output of the protocol matches the function ( $f$ ) evaluation on the private inputs of the parties.
- *Privacy* of a protocol is guaranteed if the adversary learns nothing more than the inputs and outputs of the corrupt parties, during the run of the protocol.
- *Robustness* of protocol ensures that a set of corrupt parties do not gain any influence by deviating from the prescribed protocol i.e. any influence that a set of corrupt parties gain by deviating from the protocol, can also be achieved without any deviation from the prescribed protocol.

A protocol is said to be secure in the semi-honest model, if it is correct and private. Further, a protocol is said to be information theoretic secure if it is secure in the presence of an adversary with unbounded computation power.

The proposed protocol only assumes the existence of an extended arithmetic black-box, which is characterized by a field  $\mathbb{F}_p$ , and allows a set of parties to perform the following operations securely:

- Store :  $[a] \leftarrow_P a$  i.e. party  $P$  stores a secret value  $a$  into the arithmetic black-box. The square bracket notation signifies that the element is stored in the arithmetic black-box.
- Addition :  $[c] \leftarrow [a] + [b]$  i.e.  $c$  contains the sum  $(a+b)$ .
- Multiplication :  $[c] \leftarrow [a] * [b]$  i.e.  $c$  contains the product  $(a * b)$ .
- Comparison :  $[c] \leftarrow [a] ? [b]$ , where  $c$  contains 1 if  $(a > b)$  else  $c$  contains 0.
- Equality :  $[c] \leftarrow [a] ? [b]$ , where  $c$  contains 1 if  $a$  equals  $b$ , else  $c$  contains 0.
- Release :  $a \leftarrow [a]$  implies that the stored value  $a$  is released in public.

The ideal functionality with the above six operations is termed as the “extended” arithmetic black-box, also represented as  $\mathcal{F}_{ABB}$ . Depending on the adversarial model under consideration and the efficiency requirements, one can select

an appropriate implementation of  $\mathcal{F}_{ABB}$ ; few of its implementations include [6, 8, 11, 3].

For proving the security of the proposed protocol, we employ the Universally Composability (UC) theorem stated in [7]. Roughly speaking, UC theorem states that if a protocol  $\pi_A$  is a secure implementation of its ideal functionality  $F_A$  in the presence of an ideal implementation  $F_B$  and if  $\pi_B$  securely implements  $F_B$ , then  $\pi_A$  is a secure implementation of  $F_A$  even in the presence of  $\pi_B$  rather than  $F_B$ .

For an  $n \times n$  matrix  $A$ , the notation  $[A]$  signifies that all the entries of the matrix are stored in the extended arithmetic black-box. The release operation  $A \leftarrow [A]$  signifies that each entry of the matrix  $A$  is released in public.

An adjacency matrix  $A = (a_{ij})_{n \times n}$  can also be represented as a vector of adjacency lists, i.e.,  $A = (v_i)_{n \times 1}$ , where  $v_i$  is the  $i^{th}$  adjacency vector of  $A$  or the  $i^{th}$  row of matrix  $A$ .

## 4. THE PROPOSED PROTOCOL

In this section, we provide a protocol for securely computing a random isomorphic unlabeled version of a network distributedly held by a set of parties. The graph to be constructed may be distributedly held by the  $n$  parties in various forms. For example, each party may hold a row of the adjacency matrix as her private information. Such scenarios may arise in the case of trust networks, enmity networks and sexual networks, where each individual is aware of only her outgoing links as her private information. It may also be the case that each party holds information about a subgraph in the network. Such scenarios may arise in the case of distributed social networks and financial networks. A financial network is distributedly held between a set of banks, such that each bank has information on a set of nodes/edges in the network. The details of these various forms of inputs are discussed in [15]. In this paper, we assume that each party  $P_i$  possesses an adjacency vector  $v_i$  as her private input. This protocol can easily be extended for other input forms of parties, using the functionalities available in [15].

Further we discuss the *isomorphic\_graph\_construction()* protocol proposed in the current paper. The protocol initiates by constructing  $[A]$  i.e. storing the adjacency matrix in  $\mathcal{F}_{ABB}$  through steps 1-3. In steps 4-14, a unique random number  $r_i$  is assigned to each party  $P_i$ , for all  $1 \leq i \leq n$ . The  $r_i$  values are stored in  $\mathcal{F}_{ABB}$ , such that no party learns them. The above step is implemented by assigning a random number to each party, and then checking if any two random numbers match. If yes, we repeat this procedure until we find a unique sequence of random numbers. In steps 15-19, we calculate a random permutation  $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n)$  of the set  $\{1, 2, \dots, n\}$ . We do so by assigning  $\sigma_i$  as the cardinality of the set  $\{r_j | r_i > r_j, 1 \leq j \leq n\}$ . Next we permute the adjacency matrix  $A$  to construct another matrix  $A''$ , such that  $(i, j)^{th}$  entry of matrix  $A$  is set as the  $(\sigma_i, \sigma_j)^{th}$  entry of  $A''$ . We do so by constructing an intermediary matrix  $A'$  from  $A$ , which in turn helps in constructing the matrix  $A''$ . In steps 21-24, we construct the matrix  $A'$  from  $A$ , such that  $(i, j)^{th}$  entry of matrix  $A$  is set as the  $(\sigma_i, j)^{th}$  entry of  $A'$ . In steps 25-28 we construct the adjacency matrix to be output  $A''$  from the matrix  $A'$  using a column shuffle operation i.e.  $(\sigma_i, j)^{th}$  entry of matrix  $A'$  is set as the  $(\sigma_i, \sigma_j)^{th}$  entry of  $A''$ . Hence,  $A'$  is obtained by permuting the rows of  $A$ , while  $A''$  is obtained by permuting the columns of  $A'$ . Here, both the row and column permutations

are with respect to  $\sigma$ .

---

### Protocol 1 *isomorphic\_graph\_construction()*

---

```

1: for  $i = 1$  to  $n$  do
2:   for  $j = 1$  to  $n$  do
3:      $[a_{ij}] \leftarrow_{P_i} a_{ij}$ 
4:   for  $i = 1$  to  $n$  do
5:     for  $j = 1$  to  $n$  do
6:        $[r_{ij}] \leftarrow_{P_i} r_{ij}$ 
7:   for  $i = 1$  to  $n$  do
8:      $[r_i] \leftarrow \sum_{j=1}^n [r_{ji}]$ 
9:    $[flag] \leftarrow 0$ 
10:  for  $i = 1$  to  $n - 1$  do
11:    for  $j = i + 1$  to  $n$  do
12:       $[flag] \leftarrow [flag] + ([r_i] \stackrel{?}{=} [r_j])$ 
13: if  $flag \neq 0$  then
14:   goto Step 4
15: for  $i = 1$  to  $n$  do
16:    $[\sigma_i] \leftarrow 1$ 
17: for  $i = 1$  to  $n$  do
18:   for  $j = 1$  to  $n$  do
19:      $[\sigma_i] \leftarrow [\sigma_i] + ([r_i] \stackrel{?}{>} [r_j])$ 
20:  $[A'] \leftarrow [0]_{n \times n}, [A''] \leftarrow [0]_{n \times n}$ 
21: for  $i = 1$  to  $n$  do
22:   for  $j = 1$  to  $n$  do
23:     for  $k = 1$  to  $n$  do
24:        $[a'_{jk}] \leftarrow [a'_{jk}] + ([\sigma_i] \stackrel{?}{=} j) * [a_{ik}]$ 
25: for  $i = 1$  to  $n$  do
26:   for  $j = 1$  to  $n$  do
27:     for  $k = 1$  to  $n$  do
28:        $[a''_{kj}] \leftarrow [a''_{kj}] + ([\sigma_i] \stackrel{?}{=} j) * [a'_{ki}]$ 
29:  $A'' \leftarrow [A'']$ 

```

---

The running time of the protocol depends on the number of times the goto statement in step 14 of the protocol is executed. To analyze the same, we define a random variable  $X$  to represent the number of times the goto step is executed. Let  $\exp$  represent the exponential operator.

LEMMA 1.  $E[X] \leq \exp(n^2/p)$

PROOF. The random variable  $X$  is a geometric random variable with the probability of success equal to  $\prod_{i=1}^{n-1} (1-i/p)$ .

$$\begin{aligned} \implies E[X] &= \left( \prod_{i=1}^{n-1} \left( 1 - \frac{i}{p} \right) \right)^{-1} \\ \implies E[X] &\leq \left( 1 - \frac{n}{p} \right)^{-n} \\ \implies E[X] &\leq \exp(n^2/p) \end{aligned}$$

□

Hence, the expected number of times that the goto step is executed can be made smaller than any constant number. This is achievable by using a sufficiently large prime number  $p$ , for a given  $n$ . The proposed protocol makes the above assumption regarding the field size  $p$ .

Next we analyze the distribution of the permutation  $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n)$  generated by reassigning labels to all the

vertices in the network. Let  $S_n$  represent the permutation set consisting of all  $n!$  permutations of the set  $\{1, 2, \dots, n\}$ .

**LEMMA 2.**  $\sigma \in_R S_n$  i.e.  $\sigma$  is a randomly generated permutation of the set  $\{1, 2, \dots, n\}$ .

**PROOF.** This follows directly from the fact that the sequence of numbers  $(r_1, r_2, \dots, r_n)$  are guaranteed to be unique random numbers and by the definition of  $\sigma_i$  as the cardinality of the set  $\{r_j | r_i > r_j, 1 \leq j \leq n\}$ .  $\square$

**THEOREM 1.** *The proposed isomorphic graph construction protocol is secure under the same adversarial model as the implementation of the  $\mathcal{F}_{ABB}$  functionality.*

**PROOF.** The correctness of this protocol follows from Lemma 2 and the fact that in steps 21-24 we obtain a row permuted matrix  $A'$  and in steps 25-29 we column permute  $A'$  to obtain the isomorphic network  $A''$  with respect to the permutation  $\sigma$ . The proposed protocol inherits privacy and robustness from the  $\mathcal{F}_{ABB}$  functionality as no intermediary values are revealed. It also follows from the UC theorem [7], given that the structure of the designed protocol lies within the UC framework.  $\square$

The computation cost for constructing a random isomorphic network in a non-secure manner for a given  $n$  node network would be at least  $\Theta(n^2)$ . This is because we would need to access each entry of the adjacency matrix, at least once, for constructing a random isomorphic version of it. As shown below, our protocol for computing an isomorphic version uses  $\Theta(n^3)$  operations, which has an extra factor of  $n$  compared to the non-secure variant.

**THEOREM 2.** *The proposed isomorphic graph construction protocol on an average uses  $\Theta(n^3)$  operations of the extended arithmetic black-box  $\mathcal{F}_{ABB}$ .*

**PROOF.** In steps 1-3, the protocol uses  $\Theta(n^2)$  store operations. For a sufficiently large field size  $p$ , the steps 4-14 are executed a constant number of times. Hence, the protocol uses  $\Theta(n^2)$  store operations,  $\Theta(n^2)$  addition operations and  $\Theta(n^2)$  equality checks in steps 4-14. In steps 15-19, the protocol uses  $\Theta(n)$  store operations,  $\Theta(n^2)$  addition operations and  $\Theta(n^2)$  comparison operations. Finally, in steps 20-29, the protocol employs  $\Theta(n^2)$  store and release operations and  $\Theta(n^3)$  addition, multiplication and equality operations. The theorem follows.  $\square$

## 5. CONCLUSION

In this paper, we propose a multiparty computation protocol for securely constructing an unlabeled random isomorphic version of a graph that is distributedly held by a set of  $n$  parties. The proposed protocol is proven to be secure in the  $\mathcal{F}_{ABB}$  hybrid model. The protocol can be used to study the behavioral aspects of individuals, while guaranteeing the privacy of their sensitive data. Before releasing sensitive data in public, the data is generally anonymized. The current work performs naive anonymization, on a distributedly held network, without the use of a trusted third party. One can further implement multiparty computation protocols for network specific anonymization techniques.

## 6. REFERENCES

- [1] A. Aly, E. Cuvelier, S. Mawet, O. Pereira, and M. Van Vyve. Securely solving simple combinatorial graph problems. In *Financial Cryptography and Data Security*, pages 239–257. Springer, 2013.
- [2] A. Aly and M. Van Vyve. Securely solving classical network flow problems. In *Information Security and Cryptology-ICISC*, pages 205–221. Springer, 2014.
- [3] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. of the 20th ACM STOC*, pages 1–10. ACM, 1988.
- [4] M. Blanton, A. Steele, and M. Alisagari. Data-oblivious graph algorithms for secure computation and outsourcing. In *Proc. of the 8th ACM SIGSAC AsiaCCS*, pages 207–218. ACM, 2013.
- [5] J. Brickell and V. Shmatikov. Privacy-preserving graph algorithms in the semi-honest model. In *Advances in Cryptology-ASIACRYPT*, pages 236–252. Springer, 2005.
- [6] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In *Proc. of the 20th ACM STOC*, pages 11–19. ACM, 1988.
- [7] R. Cramer, I. B. Damgård, et al. *Secure Multiparty Computation*. Cambridge University Press, 2015.
- [8] I. Damgård, M. Fitzi, E. Kiltz, J. B. Nielsen, and T. Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In *Theory of Cryptography*, pages 285–304. Springer, 2006.
- [9] K. B. Frikken and P. Golle. Private social network analysis: How to assemble pieces of a graph privately. In *Proc. of the 5th ACM WPES*, pages 89–98. ACM, 2006.
- [10] L. Garton, C. Haythornthwaite, and B. Wellman. Studying online social networks. *Journal of Computer-Mediated Communication*, 3(1):0–0, 1997.
- [11] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proc. of the 19th ACM STOC*, pages 218–229. ACM, 1987.
- [12] S. Helleringer and H.-P. Kohler. Sexual network structure and the spread of hiv in africa: evidence from likoma island, malawi. *Aids*, 21(17):2323–2332, 2007.
- [13] P. Hu, S. S. Chow, and W. C. Lau. Secure friend discovery via privacy-preserving and decentralized community detection. *arXiv preprint arXiv:1405.4951*, 2014.
- [14] V. B. Kukkala, S. Iyengar, and J. S. Saini. Secure multiparty graph computation. In *COMSNETS*, pages 1–2. IEEE, 2016.
- [15] V. B. Kukkala, J. S. Saini, and S. Iyengar. Privacy preserving network analysis of distributed social networks. *Cryptology ePrint Archive*, Report 2016/427, 2016. <http://eprint.iacr.org/2016/427>.
- [16] A. C. Yao. Protocols for secure computations. In *FOCS*, pages 160–164. IEEE, 1982.
- [17] Y. Zheng, B. Wang, W. Lou, and Y. T. Hou. Privacy-preserving link prediction in decentralized online social networks. In *Computer Security-ESORICS*, pages 61–80. Springer, 2015.

# Energy-Aware Scheduling of Periodic Conditional Task Graphs on MPSoCs

Umair Ullah Tariq and Hui Wu

The University of New South Wales

tariqu@cse.unsw.edu.au, huiw@unsw.edu.au

## Abstract

We investigate the problem of scheduling a set of periodic conditional task graphs of non-preemptible tasks on a MPSoCs (Multi Processor System-on-Chip) with shared memory such that the total expected processor energy consumption of the tasks in each scenario is minimized under two power models, namely dynamic and static power model, and propose a novel offline scheduling approach. Our approach consists of a novel two-phase task scheduler that aims at minimizing total worst-case utilization of each processor and an optimal task execution speed selection algorithm using convex NLP (Non-Linear Programming). Furthermore, we propose an  $O(1)$  time online DVS (Dynamic Voltage Scaling) heuristic that assigns each task a speed online. Our experimental results show that our two-phase scheduler achieves a 95.2% success rate of constructing a feasible schedule, compared to a 42% success rate of the state-of-the-art. For energy saving, our offline scheduling approach achieves an average improvement, a maximum improvement and a minimum improvement of 8.03%, 14.08%, and 4.8%, respectively over our online DVS heuristic.

**Keywords** Periodic conditional task graphs; energy-aware task scheduling; conditional precedence constraints, dynamic voltage scaling

## 1. Introduction

A real-time embedded system typically consists of a set of tasks. Tasks may be subject to timing constraints and precedence constraints. Precedence constraints specify data dependency and control dependency between tasks. Traditionally, precedence constraints are not subject to any conditions. However, in many applications, there are conditional precedence constraints. For example, after a task  $v_i$  completes, it may fork one of the several tasks depending on a specific condition. With conditional precedence constraints, the number of scenarios may grow exponentially with the number of conditions in the conditional tasks graph. Therefore, scheduling conditional task graphs is much more difficult than scheduling non-conditional task graphs.

Multi-core architectures such as MPSoC have been received intensive interests in the embedded systems community due to its high performance and low power. Energy minimization is one of the primary design goals for modern high performance embedded systems. Low energy consumption not only reduces heat but also increases system reliability [4]. DVS (Dynamic Voltage Scaling) is a powerful technique for reducing the energy consumption of embedded systems by dynamically changing the voltage and speed.

We investigate the problem of scheduling periodic conditional task graphs of non-preemptible tasks on a MPSoC of identical processors with continuous adjustable supply voltage and shared

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ICDCN '17, January 04–07, 2017, Hyderabad, India.

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00.  
DOI: <http://dx.doi.org/10.1145/3007748.3007782>

memory such that the total expected processor energy consumption of the tasks is minimized under two power models, namely dynamic power model and total power model, and make the following major contributions. Firstly we propose a novel two-phase offline scheduling approach that constructs a single global schedule for all the scenarios, and a speed assignment algorithm by using convex NLP (Non-Linear Programming). Furthermore, we propose an  $O(1)$  time online DVS heuristic that assigns each task a speed online to reduce the energy consumption of the task. To the best of our knowledge, this is the first work that investigates this problem. Lastly we have evaluated the performance of our offline scheduling approach using a set of benchmarks by using two performance metrics, namely schedulability and energy saving. For schedulability, the experimental results show that our approach achieves a success rate of 95.2% while the state-of-the-art approach proposed in [1] has a success rate of 42% for finding a feasible schedule. For energy saving, the experimental results show that our offline scheduling approach achieves an average improvement, a maximum improvement and a minimum improvement of 8.03%, 14.08%, and 4.8%, respectively over our online DVS heuristic.

## 2. Related Work

Only a few approaches have been proposed aiming at minimizing the energy consumption of tasks with conditional precedence constraints. Shin et al. [9] propose a scenario-based offline NLP that assigns each task different speeds in different scenarios. The approach has exponential complexity as it constructs unique schedule for each scenario. In [5, 7] online algorithms are proposed that assign each task a speed based on the length of critical path. However these approaches have an exponential time complexity in the worst case since they enumerate all the possible scenarios when computing the critical path lengths. A sporadic conditional DAG task model is introduced in [2] which specifies each DAG as a 3 tuple  $(G(V_i, E_i), D_i, T_i)$ , where  $G(V_i, E_i)$  is a DAG containing conditional vertices,  $D_i$  is the deadline and  $T_i$  is the period. Furthermore, it is shown that the tight speed-up bound for global EDF (Earliest Deadline First) found in [3] still holds for conditional sporadic task model. In [8] efficient schedulability tests are proposed for GEDF (Global Earliest Deadline First), and it is shown that the response time analysis presented in [8] outperforms the analysis in [2]. In [1] federated scheduling is proposed that assigns a set of processors exclusively to heavy conditional task graphs and the remaining processors to light conditional task graphs. However, the approach has a low success rate in terms of finding a feasible schedule.

## 3. System and Task Models

The target MPSoC consists of a set  $P = \{pe_1, pe_2, \dots, pe_m\}$  of  $m$  identical processors. All the processors have a shared memory. Therefore, the communication time between tasks is negligible. Each processor  $pe_i$  is dynamic voltage and frequency scaling (DVFS) enabled. We assume that each processor can operate in a continuous frequency range  $[f_{min}, f_{max}]$ . The total power consumption of a processor consists of dynamic power due to switching activity, and static power as a result of leakage. The total power can be calculated as  $P_{tot} = C_{eff}V_{dd}^2f + L_g(V_{dd}K_3e^{K_4V_{dd}}e^{K_5V_{bs}} + |V_{bs}|I_j)$  [4], where  $C_{eff}V_{dd}^2f$  is dynamic power,  $L_g$  is the number of logic gates in the circuit,  $K_3$ ,  $K_4$  and  $K_5$  are the processor technology dependent parameters, and  $V_{bs}$  and  $I_j$  are the body-bias voltage and body junction leakage current, respectively.

We consider a set  $\Gamma = \{\tau_1, \tau_2, \dots, \tau_n\}$  of  $n$  independent periodic applications. Each application  $\tau_i \in \Gamma$  is described by a 3-tuple  $(G_i, D_i, T_i)$ , where  $G_i = (V_i, E_i, A_i)$  is a CTG (Conditional Task Graph),  $D_i$  and  $T_i$  are the relative deadline and period, respectively, of  $\tau_i$ . We assume that  $D_i \leq T_i$  holds. A CTG  $G_i = (V_i, E_i, A_i)$  is a directed acyclic graph, where  $V_i = \{v_{i,1}, v_{i,2}, \dots, v_{i,n_i}\}$  is a set of tasks,  $E_i \subseteq V_i \times V_i$  is a set of directed edges each denoting the dependency between the two tasks, and  $A_i$  is a set of triplets  $(e_{s,t}, c_{s,t}, p(c_{s,t}))$ , where  $e_{s,t} \in E_i$ , and  $c_{s,t}$  and  $p(c_{s,t})$  represent the condition associated with  $e_{s,t}$  and its probability, respectively. We assume that each  $G_i$  has a single source node  $v_{i,\text{source}}$  and a single sink node  $v_{i,\text{sink}}$ . Furthermore, each task  $v_{i,j}$  in a CTG  $G_i$  has a WCET (Worst-Case Execution Time), denoted by  $w_{i,j}$ .

In a CTG, an edge is called a conditional edge if it is associated with a condition representing that the following task is executed only if the condition holds. Otherwise, it is called an unconditional edge. Each non-sink node of a CTG is a FORK node. A FORK node with multiple outgoing edges is an OR-FORK node if all its outgoing edges are conditional edges with mutually exclusive conditions. The sum of the probabilities of all the conditional edges of an OR-FORK node is equal to one. A node is an AND-FORK node if all its outgoing edges are unconditional edges. A non-source node is also a JOIN node. A JOIN node with multiple incoming edges is an OR-JOIN node if all its parents are mutually exclusive. A node is an AND-JOIN node if all its parent tasks are executed. OR-FORK and OR-JOIN nodes always appear in a pair and are referred to as conditional constructs. The restrictions imposed on the edges of CTGs are the same as in [2, 8].

A scenario of a CTG  $G_i$  is a sub-graph of  $G_i$  formed by all the tasks in a complete execution trace of the task set. Given a CTG  $G_i$ , an activation space  $AS_i$  is a set of all the possible conditions each of which corresponds to a unique scenario. Associated with each task is its activation probability. The activation probability of a task is the probability with which the task is executed. Let  $S_j$  be the set of scenarios which a task  $v_j$  belongs to. The activation probability of  $v_j$  is calculated as  $p(v_j) = \sum_{s \in S_j} p(s)$ , where  $p(s) = \prod_{c \in s} p(c)$ . The activation probability can be computed in polynomial time using the algorithm proposed in [6].

## 4. Offline Scheduling

### 4.1 Conditional Task Graph Priority

Each CTG  $G_i$  in set  $\Gamma$  is assigned a priority  $\omega_i = \frac{\max\{\zeta_{ij}; v_{ij} \in V_i\}}{T_i}$ , where  $\zeta_{ij}$  is the finish time of each task  $v_{ij}$  when  $G_i$  is scheduled alone on the target MPSoC. A higher value of  $\omega_i$  implies a higher priority. To compute  $\omega_i$ , each CTG is individually scheduled using the single CTG scheduling algorithm presented in [10].

### 4.2 Task Mapping and Local Scheduling

The task mapping and local scheduling algorithm aims at minimizing the total worst-case utilization of all the processors so that the workload across all processors is balanced. Let  $L_i^k$  be the set of tasks in  $\tau_i$  scheduled on  $pe_k$ . The worst-case utilization of  $pe_k$  is defined as  $\mu^k = \sum_{\tau_i \in \Gamma} \frac{\max_{s \in AS_i} \{\sum_{v_{ij} \in s \cap L_i^k} w_{i,j}\}}{T_i}$ . A brute-force algorithm for computing the worst-case utilization of  $pe_k$  has exponential time complexity. Let  $S_i^k$  be the set of tasks of  $G_i$  assigned to  $pe_k$  such that the sum of worst case execution times of all the tasks in the set is maximized in all the possible scenarios of  $G_i$ , and  $\Gamma^k$  be the set of all the CTGs having at least one task assigned to  $pe_k$ . The worst case utilization can be calculated as follows:  $\mu^k = \sum_{S_i^k \in S^k} \frac{\sum_{v_{ij} \in S_i^k} w_{i,j}}{T_i}$ , where  $S^k = \bigcup_{\tau_i \in \Gamma^k} \{S_i^k\}$ .

Algorithm 2 computes the  $S_i^k$  by computing the partial worst-case set  $WCS_{ij}^k$  of each task  $v_{ij} \in V_i$  in reverse topological order. We define the worst-case set of a task  $v_{i,j} \in V_i$  with respect to a processor  $pe_k$ , denoted by  $WCS_{i,j}^k$ , as follows:

DEFINITION 1. Consider the following cases:

---

#### ALGORITHM 1: Task Mapping and Local Scheduling

---

```

input : A set  $\Gamma$  of periodic applications, a MPSoC with  $m$  processors, a set  $\Omega$  of priorities of all  $\tau_i \in \Gamma$ .
output: Schedule graph  $G_s(V, E)$  and a schedule
1 . for each  $pe_k \in P$  do
2   |   $S^k \leftarrow \emptyset$ 
3   |   $V \leftarrow \emptyset; E \leftarrow \emptyset; A \leftarrow \emptyset;$ 
4   |  while there are unscheduled CTGs in  $\Gamma$  do
5     |    Select  $\tau_i \in \Gamma_1$  that has the maximum value of  $\omega$ ;
6     |    Compute the successor tree consistent deadline of each
7     |     $v_{i,j} \in V_i$ ;
8     |    Construct a list  $\Theta$  of all the tasks  $v_{ij} \in V_i$  sorted in
9     |    non-increasing order of successor tree consistent deadlines;
10    |    for each  $v_{i,j} \in \Theta$  from source to sink do
11      |      for each  $pe_k \in P$  do
12        |        Construct a set  $L_i^k$  of all the tasks of CTG  $\tau_i$  assigned
13        |        to processor  $pe_k$ ;
14        |         $L_i^k \leftarrow (L_i^k \setminus \text{Mutex}_{i,j}) \cup \{v_{i,j}\}$ ;
15        |        Find the previous  $S_i^k \in S^k$  and assign to  $temp^k$ ;
16        |         $S_i^k \leftarrow \text{worst\_case\_set}(L_i^k, \Theta)$ ;
17        |         $S^k \leftarrow (S^k \setminus \{temp^k\}) \cup \{S_i^k\}$ ;
18        |         $\mu^k = \sum_{S_i^k \in S^k} \frac{\sum_{v_{i,l} \in S_i^k} w_{i,l}}{T_i}$ ;
19        |         $S^k \leftarrow (S^k \setminus \{S_i^k\}) \cup \{temp^k\}$ ;
20        |      Find  $pe_k$  that has the minimum worst case utilization  $\mu^k$ 
21        |      and assign  $v_{i,j}$  to  $pe_k$ ;
22        |       $S^k \leftarrow (S^k \setminus \{temp^k\}) \cup \{S_i^k\}$ ;
23        |      for  $u = 1$  to  $\frac{H}{T_i}$  do
24          |         $r_{i,j,u} = \max\{(u-1).T_i, \max\{\zeta(v_{i,l,u}) : v_{i,l,u} \in IPred(v_{i,j,u})\}$ ;  $d_{i,j,u} = (u-1)T_i + D_i$ ;
25          |         $\rho_{i,u,j} \leftarrow \text{start\_time}(v_{i,j}, r_{i,u,j}, d_{i,u,j}, pe_k)$ ;
26          |         $\zeta_{i,j,u} \leftarrow \rho_{i,j,u} + w_{i,j,u}$ ;  $V \leftarrow V \cup \{v_{i,j,u}\}$ ;
27          |         $E \leftarrow E \cup \{(v_{i,l,u}, v_{i,j,u}) : \forall v_{i,l} \in IPred_{i,j}\}$ ;
28          |         $A \leftarrow A \cup \{(v_{i,l,u}, v_{i,j,u}) : \forall v_{i,l} \in IPred_{i,j} \wedge v_{i,l}$  is OR-FORK node  $\}$ ;
29        |      Function  $\text{start}_T(v_{i,j}, r_{i,j}, d_{i,j}, pe_k)$ 
30          |         $start \leftarrow r_{i,j}$ ;
31          |        if a task is assigned to  $pe_k$  then
32            |           $end \leftarrow \min\{\zeta_{w,l}, d_{i,j}\}$ ; // where  $v_{w,l}$  is the
33            |          latest task on  $pe_k$ 
34            |        for each  $v_{o,q}$  in increasing order of  $\rho_{o,q}$  scheduled in the
35            |        interval  $[start, end]$  of  $pe_k$  do
36              |          if  $v_{o,q} \notin \text{Mutex}_{i,j}$  &  $(start + w_{i,j}) > \rho_{o,q}$  then
37                |             $start \leftarrow \zeta_{o,q}$ ;
38            |        return  $start$ ;

```

---

- $v_{i,j}$  is a sink node. If  $v_{i,j} \notin L_i^k$  holds, we have  $WCS_{i,j}^k = \emptyset$ . Otherwise, we have  $WCS_{i,j}^k = \{v_{i,j}\}$ .
- $v_{i,j}$  is an OR-FORK node. Let  $v_{i,o}$  be a task in  $ISucc_{i,j}$  satisfying  $\sum_{v_{i,l} \in WCS_{i,o}^k} w_{i,l} = \max\{\sum_{v_{i,l} \in WCS_{i,q}^k} w_{i,l} : v_{i,q} \in ISucc_{i,j}\}$ . If  $v_{i,j} \notin L_i^k$  holds, we have  $WCS_{i,j}^k = WCS_{i,o}^k$ . Otherwise, we have  $WCS_{i,j}^k = WCS_{i,o}^k \cup \{v_{i,j}\}$ .
- $v_{i,j}$  is an AND-FORK node. If  $v_{i,j} \notin L_i^k$  holds, we have  $WCS_{i,j}^k = \bigcup_{v_{i,l} \in ISuc_{i,j}} WCS_{i,l}^k$ . If  $v_{i,j} \in L_i^k$  holds, we have  $WCS_{i,j}^k = \bigcup_{v_{i,l} \in ISuc_{i,j}} WCS_{i,l}^k \cup \{v_{i,j}\}$

Our task mapping and local scheduling algorithm, as shown in Algorithm 1, first selects the CTG  $G_i$  with the maximum value of  $\omega$ . It then works iteratively by selecting a task  $v_{i,j} \in G_i$  with the minimum successor tree deadline [10] and assigns it to the processor having the minimum total worst case utilization. Next, it generates all the jobs of  $v_{i,j}$  in hyper period  $H$  (the least common multiple of periods of all the applications). We use three subscripts to represent a job. For each job  $v_{i,j,u}$  it finds the start time  $\rho_{i,j,u}$ .

---

**ALGORITHM 2:** Computing the worst-case Scenario

---

```

1 Function worst_case_set( $L_i^k, \Theta$ )
2   for each task  $v_{i,j}$  in  $\Theta$  from sink to source do
3     if  $v_{i,j}$  is a sink node then
4        $WCS_{i,j}^k \leftarrow \emptyset$ ;
5     else if  $v_{i,j}$  is an OR-FORK node then
6       Find a child  $v_{i,l}$  of  $v_{i,j}$  among all the children of  $v_{i,j}$ 
7       in  $G_i$  such that  $\sum_{v_{i,q} \in WCS_{i,l}^k} w_{i,q}$  is maximized;
8        $WCS_{i,j}^k \leftarrow WCS_{i,l}^k$ ;
9     else
10     $WCS_{i,j}^k \leftarrow \emptyset$ ;
11    for each  $v_{i,l} \in ISuc_{i,j}$  do
12       $WCS_{i,j}^k \leftarrow WCS_{i,j}^k \cup WCS_{i,l}^k$ ;
13
14    if  $v_{i,j} \in L_i^k$  then
15       $WCS_{i,j}^k \leftarrow WCS_{i,j}^k \cup \{v_{i,j}\}$ ;
16
17  $S_i^k \leftarrow WCS_{i,source}^k$ ;
18 return  $S_i^k$ ;

```

---

**Table 1.** Characteristics of Benchmarks.

BM	x/y/z	T	D	BM	x/y/z	T	D	BM	x/y/z	T	D
ctg1	15/1/3	185 <sup>1</sup>	175	ctg2	18/1/2	185 <sup>1</sup>	176	ctg13	34/4/10	274	260
ctg3	20/3/7	370	351	ctg4	20/3/6	370	352	ctg14	35/4/10	274	260
ctg5	18/2/5	170	161	ctg6	17/1/3	170	162	ctg15	15/1/3	548	520
ctg7	17/1/3	340	323	ctg8	20/1/2	340	325	ctg16	32/2/4	548	522
ctg9	15/1/3	184	175	ctg10	18/1/2	184	176	ctg17	16/1/3	548	530
ctg11	20/3/7	368	349	ctg12	20/3/6	368	350	ctg18	34/2/5	345	339
ctg19	35/6/14	690	655	ctg20	34/3/7	690	657	ctg21	34/2/5	345	339
ctg22	20/1/2	690	680	ctg23	30/2/6	305	298	ctg24	34/2/6	305	299
ctg25	33/4/9	610	598	ctg26	32/4/10	610	597	ctg27	15/1/3	610	599
ctg28	34/3/6	324	322	ctg29	32/6/14	648	640	ctg30	32/2/4	648	643
ctg31	34/3/6	324	321								

When computing the start time for each job, Algorithm 1 takes into account the mutual exclusion relation between tasks. Two tasks are said to be mutually exclusive if there exists no scenario where they can co-exist. Set  $Mutex_{i,j}$  contains all the tasks that are mutually exclusive to task  $v_{i,j}$ . Two mutually exclusive tasks can be assigned to the same time slot on the same processor. Algorithm 1 repeats the above mentioned steps until all CTGs have been scheduled.

Algorithm 1 outputs a super graph  $G_s(V, E, A)$  that contains all the jobs of all the tasks of all the CTGs in set  $\Gamma$  and a schedule for all the jobs in the super graph. Additional precedence constraints introduced by the schedule are captured by inserting additional edges to super graph  $G_s$ . Associated with each task  $v_{i,j}$  is a set called  $cSet_{i,j}$ . This set contains all the tasks that are concurrent to task  $v_{i,j}$ . Two tasks  $v_{i,j}, v_{i,l} \in G_i$  are said to be concurrent if they are not reachable from each other in  $G_i$  and are not mutually exclusive  $v_{i,l} \notin Mutex_{i,j}$ . Furthermore,  $cSet_{i,j}$  contains all the tasks from other CTGs.

An edge  $(v_{i,j,u}, v_{w,o,l})$  is inserted between the jobs  $v_{i,j,u}$  and  $v_{w,o,l}$  if the following three constraints hold simultaneously. Firstly,  $v_{i,j,u}$  and  $v_{w,o,l}$  are scheduled on same processor. Secondly, the start time of task  $v_{w,o,l}$  must be greater than the start time of  $v_{i,j,u}$ . Lastly,  $v_{w,o} \in cSet_{i,j}$  holds,  $v_{i,j}$  and  $v_{w,o}$  are concurrent tasks.

## 5. Task Speed Assignment

We propose two approaches to assign a speed to each job. The first approach is an online DVS algorithm that assigns each job a speed at runtime. The second approach uses NLP to assign each job an optimal speed off-line.

### 5.1 Online DVS Algorithm

Our online DVS algorithm works as follows:

- Calculate the slack available for  $v_i$ :  $Slack = ECD_i - CurrTime - w_i$
- Calculate the MultRatio for  $v_i$ :  $MulRatio \leftarrow \frac{slack+CP_i}{CP_i}$ .

- Calculate the frequency  $f_i$  for  $v_i$ :  $f_i \leftarrow \frac{NCC_i}{w_i \times MulRatio}$

The online algorithm determines the available slack for job  $v_i$  using  $ECD_i$  (edge consistent deadline) and  $CurrTime$  (the time when online algorithm is called). It determines the amount of slack to allocate to each job using  $CP_i$  the critical path length of job  $v_i$ . The parameters  $ECD_i$  and  $CP_i$  are computed offline as follows:

- Perform topological sort on  $G_s$ .
- For every job  $v_i \in G_s$  in topological order, calculate its edge consistent release time  $ECR_i \leftarrow \max\{r_i, \max\{ECR_j + w_j : \forall v_j \in IPred_i\}\}$ .
- For every job  $v_i \in G_s$  in reverse topological order, calculate the edge consistent deadline  $ECD_i \leftarrow \min\{d_i, \min\{ECD_j + w_j : \forall v_j \in IPred_i \wedge ECR_j < ECD_i\}\}$ .

Clearly, our online algorithm takes  $O(1)$  time.

## 5.2 NLP Approach

The task speed assignment problem is solved by converting the problem into a convex NLP problem, aiming at minimizing the expected energy consumption while satisfying the precedence and timing constraints. The energy consumed by a task  $v_i$  running at frequency  $f_i$  is given as  $E_i = P(f_i)et_i$ , where  $et_i$  is the execution time to task  $v_i$  and  $P(f_i)$  is either the dynamic power function or the total power function depending on if we aim at minimizing the total dynamic processor energy or the total processor energy. Under a specific power model, the expected total energy consumption of super CTG  $G_s$  is  $\sum_{v_j \in V} E_j p(v_j)$ <sup>1</sup>.

The super graph  $G_s$  may contain redundant edges, introducing redundant constraints in the NLP formulation. A reduced super graph  $G_R$  is constructed from the super graph  $G_s(V, E, A)$  by applying transitive reduction to  $G_s$ . Given the reduced graph  $G_R(V, E, A)$ , we formulate the task speed assignment problem as follows:

$$\begin{aligned} \min \quad & \sum_{v_j \in V} E_j p(v_j) \quad s.t: \\ & \forall v_j, et_j = NCC_j \frac{K_6 L_d V_{dd_j}}{((1 + K_1)V_{dd_j} + K_2 V_{bs} - V_{th_1})^\alpha} \\ & \forall (v_j, v_l) \in E_R, \rho_j + et_j \leq \rho_l \\ & \forall v_j \in V, \rho_j + et_j \leq d(v_j) \\ & \forall v_j \in V, \rho_j \geq r(v_j) \\ & V_{dd_{min}} \leq V_{dd_j} \leq V_{dd_{max}} \end{aligned}$$

where Equations (2), (3), (4) are the precedence, deadline and release time constraints respectively. By solving this NLP problem, we can find for each task  $v_j$ , the voltage  $V_{dd_j}$ , its start time  $\rho_j$  and execution time  $et_j$  such that the expected energy consumption of CTG  $G_R$  is minimized under a specific power model.

Notice that all the constraints and the objective function are convex. Therefore, this NLP problem can be solved in polynomial time [11].

## 6. Experimental Results

### 6.1 Experimental Setup

In our experiments, we consider 70 nm technology. The values of parameters for 70 nm are taken from [4]. We have implemented our approach and the federated scheduling approach proposed in [1] using Matlab version R2015a and use the fmincon solver to solve NLP problems. The hardware platform consists of Intel(R) Core(TM) i5-4570 CPU with a clock frequency of 3.20 GHz, 8.00 GB memory and 3 MB cache. The details of all the benchmarks are given in Table 1, where x, y, z, BM, T and D stands for the number of tasks, the number of OR-FORK tasks and the number

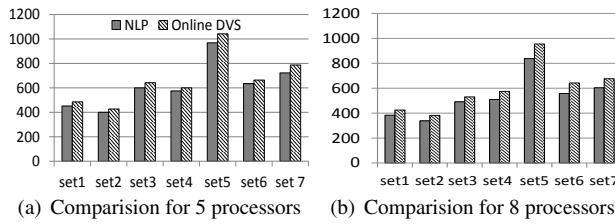
<sup>1</sup> Energy can be expressed in terms of supply voltage completely since  $et_i = \frac{NCC_i}{f_i}$ , where  $NCC_i$  is worst case clock cycles of  $v_i$ .

**Table 2.** Comparison of approach in [1] and proposed approach

Set	# pe 4		# pe 5		# pe 8	
	Schd [1]	Schd Ours	Schd [1]	Schd Ours	Schd [1]	Schd Ours
1	No	Yes	Yes	Yes	Yes	Yes
2	No	Yes	Yes	Yes	Yes	Yes
3	No	No	No	Yes	Yes	Yes
4	No	Yes	No	Yes	Yes	Yes
5	No	Yes	No	Yes4	Yes	Yes
6	No	Yes	No	Yes	Yes	Yes
7	No	Yes	No	Yes	Yes	Yes

**Table 3.** CTGs Sets.

Set	Conditional Task Graphs
1	ctg1, ctg2, ctg3, ctg4
2	ctg5, ctg6, ctg7, ctg8
3	ctg9, ctg10, ctg11, ctg12
4	ctg13, ctg14, ctg15, ctg16, ctg17
5	ctg18, ctg19, ctg20, ctg21, ctg22
6	ctg23, ctg24, ctg25, ctg26, ctg27
7	ctg28, ctg29, ctg30, ctg31

**Figure 1.** Comparison of the NLP approach and the online DVS approach for two variations of processors (y-axis represents average energy consumption in milli joule)

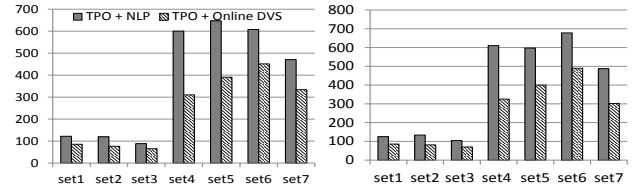
of conditions, benchmark, the period and deadline respectively, in each benchmark. The benchmarks listed in Table 1 are the same benchmarks used in [6].

Based on the benchmarks, we have formed seven sets of CTGs as shown in Table 3.

## 6.2 Results and Discussion

In the first set of experiments, we compare online DVS algorithm with NLP. We use average energy consumption which is the sum of energy consumption of all the scenarios divided by total number of scenarios as a metric for comparison. Since NLP cannot claim slack available at runtime, we assume that each job takes worst case execution time to execute. Figures 1(a) and 1(b) give a comparison of average energy consumptions of the NLP approach and the online DVS algorithm on seven sets of conditional task graphs shown in Table 3. Compared to the online DVS algorithm, the NLP approach achieves an average improvement 8.03%. Furthermore, it achieves a maximum improvement of 14.08% for set 6 on 8 processor and minimum improvement of 4.8% for set 2 on 4 processors. Compared to the online DVS algorithm, the NLP approach takes much longer time to solve. Figures 2(a) and 2(b) give a comparison of times taken by the NLP approach and the online DVS algorithm to solve a given problem. The overhead of the online DVS algorithm on average is only 3 ms for one complete scenario. In case of the online DVS algorithm, the total time taken by the two-phase offline scheduler and the online DVS algorithm to solve the problem on average is 253 ms. In contrast, the total time taken by the two-phase offline scheduler and NLP is 387 seconds.

We have also conducted a second set of experiments to compare our approach with the state-of-the-art presented in [1]. We observe in our experiments that assigning processors exclusively to heavy CTGs results in lower processor utilization. Furthermore, we observe that restricting an entire CTG to run on a single core, unnecessarily delays tasks, which results in a low success rate of constructing a feasible schedule. In Table 2, the feasible schedules (Schd [1]) constructed by the approach proposed in [1] is much lower compared to our approach. The success rate of the federated scheduling approach for all the problem instance is 42%, compared to a success rate of 95.2% for our approach.

**Figure 2.** Comparison of our approach and the federated scheduling approach for two variations of processors (y-axis represents time in seconds)

## 7. Conclusion

We have proposed a novel two-phase offline approach to the problem of scheduling a set of periodic conditional task graphs on  $m$  identical DVS-enabled processors with shared memory. Our approach consists of a task mapping algorithm and an NLP formulation for finding the optimal speed of each task such that the expected total processor energy consumption of all the tasks is minimized. Furthermore, we have proposed an online DVS scheduling heuristic. We have shown that our offline scheduling approach performs significantly better than the state-of-the-art approach [1] in terms of finding a feasible schedule. Furthermore, we have compared our NLP approach with our online DVS heuristic. The NLP approach achieves, an average improvement, a maximum improvement and a minimum improvement of 8.3 %, 14.08%, and 4.8%, respectively, over our online DVS heuristic. Compared to the state-of-the-art approach without DVS, our approach without DVS has a much higher success rate for finding a feasible schedule.

## References

- [1] S. Baruah. The federated scheduling of systems of conditional sporadic dag tasks. In *Proceedings of the 12th International Conference on Embedded Software*, pages 1–10. IEEE Press, 2015.
- [2] S. Baruah, V. Bonifaci, and A. Marchetti-Spaccamela. The global edf scheduling of systems of conditional sporadic dag tasks. In *Proceedings of the 27th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 222–231. IEEE, 2015.
- [3] V. Bonifaci, A. Marchetti-Spaccamela, S. Stiller, and A. Wiese. Feasibility analysis in the sporadic dag task model. In *Proceedings of the 25th Euromicro Conference on Real-Time Systems (ECRTS)*, pages 225–233. IEEE, 2013.
- [4] G. Chen, K. Huang, and A. Knoll. Energy optimization for real-time multiprocessor system-on-chip with optimal dvfs and dpm combination. *ACM Transactions on Embedded Computing Systems (TECS)*, 13(3s):111, 2014.
- [5] Y. Ge, Y. Zhang, P. Malani, Q. Wu, and Q. Qiu. Low power task scheduling and mapping for applications with conditional branches on heterogeneous multi-processor system. *Journal of Low Power Electronics*, 8(5):535–551, 2012.
- [6] M. Lombardi, M. Milano, M. Ruggiero, and L. Benini. Stochastic allocation and scheduling for conditional task graphs in multi-processor systems-on-chip. *Journal of scheduling*, 13(4):315–345, 2010.
- [7] P. Malani, P. Mukre, Q. Qiu, and Q. Wu. Adaptive scheduling and voltage scaling for multiprocessor real-time applications with non-deterministic workload. In *Proceedings of the conference on Design, automation and test in Europe*, pages 652–657. ACM, 2008.
- [8] A. Melani, M. Bertogna, V. Bonifaci, A. Marchetti-Spaccamela, and G. C. Buttazzo. Response-time analysis of conditional dag tasks in multiprocessor systems. In *Proceedings of the 27th Euromicro Conference on Real-Time Systems (ECRTS)*, pages=211–221, year=2015, organization=IEEE.
- [9] D. Shin and J. Kim. Power-aware scheduling of conditional task graphs in real-time multiprocessor systems. In *Proceedings of the 2003 international symposium on Low power electronics and design*, pages 408–413. ACM, 2003.
- [10] U. U. Tariq and H. Wu. Energy-aware scheduling of conditional task graphs with deadlines on mpsoc. In *Proceedings of the 34th IEEE International Conference on Computer Design (ICCD)*, pages 265–272. IEEE, 2016.
- [11] A. N. Yurii Nesterov. *Interior Point Polynomial Algorithms in Convex Programming*. SIAM, 1987.

# Exact, Fast and Scalable Parallel DBSCAN for Commodity Platforms

Sonal Kumari, Poonam Goyal, Ankit Sood, Dhruv Kumar, Sundar Balasubramaniam, Navneet Goyal

Advanced Data Analytics & Parallel Technologies Laboratory

Department of Computer Science & Information Systems, BITS-Pilani, Pilani Campus, INDIA  
 (sonal.kumari, poonam, h2014197, f2010526, sundarb, goel)@pilani.bits-pilani.ac.in

## ABSTRACT

DBSCAN is one of the most popular density-based clustering algorithm capable of identifying arbitrary shaped clusters and noise. It is computationally expensive for large data sets. In this paper, we present a grid-based DBSCAN algorithm, GridDBSCAN, which is significantly faster than the state-of-the-art sequential DBSCAN. The efficiency of GridDBSCAN is achieved by reducing the number of neighborhood queries using spatial locality information, without compromising the quality of clusters. We also propose scalable parallel implementations of GridDBSCAN to leverage a multicore commodity cluster. Clustering results of GridDBSCAN and its parallel implementations are exactly the same as that of classical DBSCAN. The performance of proposed algorithms, both sequential and parallel, is benchmarked against the state-of-the-art algorithms by experimenting on various real datasets. Experimental results show considerable performance improvements achieved by GridDBSCAN and its parallel implementations.

## Categories and Subject Descriptors

- Information systems → Clustering
- Computing methodologies → Distributed algorithms
- Computing methodologies → Concurrent algorithms.

## Keywords

Parallel computing; clusters; density-based clustering; DBSCAN; R-tree; spatial locality

## 1. INTRODUCTION

Clustering is an important task in Data Mining and Machine Learning to identify disjoint groups of points that are similar. Density-based clustering is capable of forming arbitrary shaped clusters and it does not require number of clusters as input, unlike several other clustering algorithms. DBSCAN [15] is a popular density-based clustering algorithm which forms a cluster around each point having more than  $MinPts$  number of points in its neighborhood of radius  $\epsilon$ . Such points are referred to as *core points*. If the  $\epsilon$ -neighborhood of a point has fewer points than  $MinPts$ , and does not contain a core point, it is classified as a *noise point*.

DBSCAN has a complexity of  $O(n \log n)$  when implemented using spatial tree indexes [15]. It has predominantly sequential data access patterns which makes it challenging to parallelize. Many

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

ICDCN '17, January 04-07, 2017, Hyderabad, India

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007773>

existing approaches to parallelizing DBSCAN adopt the master-slave model for computation [3, 4, 9, 12, 13, 37, 38]. High communication costs between master and its slaves and limited parallelization during the merging phase at the master node renders the master-slave model inefficient. Some parallel implementations of DBSCAN on MapReduce/Hadoop are presented in [22, 38]. PDSDBSCAN [27] used union-find (*UF*) data structure to break the sequential data access in DBSCAN to make it amenable to parallelization and to achieve better scalability. The performance of the sequential DBSCAN presented in this paper is comparable to that of classical DBSCAN [15]. As a result, PDSBSCAN remains computationally expensive for large datasets. There are few parallel algorithms [29, 30, 34, 35] which achieves performance but at the cost of clustering accuracy.

In this paper, we have developed a novel version of the DBSCAN algorithm, named GridDBSCAN, which is considerably faster than classical DBSCAN. In GridDBSCAN, a virtual grid is superimposed over the data space and points belonging to cells that are dense enough are not processed for neighborhood queries. In these cases, point-wise neighborhood queries are not required thus drastically reducing the number of point-wise neighborhood queries without compromising the quality of clusters obtained. Moreover, we use a variant of *R*-tree [21] named Grid-R-tree [39] that supports efficient querying of cell-wise neighborhoods. A cell-wise neighborhood query returns all cells in the neighborhood of a given cell. Grid-R-tree stores cells at the top-level and points per cell in the next level therefore making cell-wise queries efficient. In summary, compared to classical DBSCAN, we have reduced the total number of queries, and a significant portion of these queries are cell-wise neighborhood queries thereby considerably reducing the runtime of the algorithm. GridDBSCAN runs up to 8.98x faster than the state-of-the-art implementation [27] of classical DBSCAN. This is mainly due to optimized neighborhood queries. Our experimental results show that GridDBSCAN performs at most 87% of the queries of classical DBSCAN. Approximately, 50% of these queries are cell-wise queries.

We have parallelized GridDBSCAN for shared memory and distributed memory systems. We also present the hybrid implementation of GridDBSCAN to leverage a multicore cluster system. Steps involved in parallel GridDBSCAN algorithm are: 1) data distribution using kd-tree partitioning, preserving spatial locality and ensuring equal-sized partitions, 2) local clustering using GridDBSCAN, and 3) *efficient* merging of local clusters in a tree-parallel mode.

The shared memory implementation uses local disjoint-set data structure to store clustering information to avoiding locking, thereby enhancing the efficiency and scalability of the algorithm. Moreover, the merging of local clusters does not need any neighborhood queries, resulting in minimized communication overheads in the distributed memory implementation.

We have benchmarked our proposed parallel implementations with union-find based parallel DBSCAN [27]. It is found that our algorithms have either better or comparable scalability. The execution time is much less than that of [27]. Shared memory implementation is up to 37.68x faster than GridDBSCAN using 48 number of threads and is up to 99.96x faster on 48 threads than that of [12]. Distributed implementation of GridDBSCAN is up to 40.08x faster than GridDBSCAN on 32 nodes. We have also compared it with that of the best existing distributed implementation of DBSCAN [27] and found up to 32.58x faster on 32 nodes than that of [27]. Hybrid implementation of GridDBSCAN is achieving up to 100.63x speed up with respect to GridDBSCAN on 32 nodes using 4 threads on each node.

The rest of the paper is organized as follows. Section 2 gives details of related work. In Section 3, classical DBSCAN is described. In Sections 4 and 5, we present the proposed sequential GridDBSCAN algorithm and the proposed parallelization strategies for shared, distributed, and hybrid systems, respectively. Experimental set up and results are presented in Section 6. Conclusions and future directions are given in Section 7.

## 2. RELATED WORK

DBSCAN was introduced by Ester et al. [15] which uses  $R^*$ -tree [6] for region query. Time complexity of the algorithm is  $O(n \log n)$ , where  $n$  is the number of data points to be clustered. Aoying et al. [3] gave various optimized versions of the DBSCAN algorithm including a parallel DBSCAN. The parallel DBSCAN algorithm runs local DBSCAN on multiple cores and local results are merged to get the global clustering result.

To achieve performance gain, some authors have proposed approximate parallel algorithms for DBSCAN, viz., PARDICLE [29, 30], Mr. Scan [34, 35]. PARDICLE is based on approximate neighborhood computations using sampling in high density regions. Mr. Scan [34, 35] uses representative points instead of the entire  $\epsilon$ -neighborhood. This preserves the quality of local DBSCAN but may skip merging in a few cases resulting in approximate clusters.

The master-slave computing model has been used in many attempts to parallelize DBSCAN [4, 9, 13, 37]. Xu et al. [37] gave a parallel DBSCAN, named as PDBSCAN, which uses  $dR^*$ -tree for region query.  $dR^*$ -tree is a variant of  $R^*$ -tree in which  $R^*$ -tree is replicated over multiple nodes for efficient data access on a distributed systems. The leaf nodes are distributed over multiple nodes for minimizing communication cost and load balancing. Arlia et al. [4] gave a parallel DBSCAN algorithm in which DBSCAN is divided into two major operations: clustering assignment and neighborhood querying. Master node performs clustering assignment while all slaves perform neighborhood queries in parallel. Coppola et al. [13] also gave a similar master-slave model based parallel DBSCAN in which each slave keeps a copy of an  $R^*$ -tree. The master processes the data points in the same order as classical DBSCAN, and slaves execute neighborhood queries. The major drawback of master-slave model is serialized computation at master node which affects the scalability of the parallel algorithms.

Chen et al. [12] gave a parallel DBSCAN, named as P-DBSCAN, which distributes the data among several nodes, build Priority  $R$ -tree on each node, run local DBSCAN, and aggregate the local results to get global clustering results. Priority  $R$ -tree is a variant of  $R$ -tree which performs efficient region queries. Patwary et al. [27] present a scalable DBSCAN algorithm using union-find data structure [28] to break the sequential access of data and make it suitable to high performance computing. The performance of

union-find based sequential DBSCAN [27] and classical DBSCAN is comparable and thus local computations are inefficient. The union-find based sequential DBSCAN is parallelized for both distributed and shared memory systems.

Some MapReduce and GPU-based implementations of DBSCAN [1, 11, 22, 23, 34, 35] also exist in literature. MR-DBSCAN [22, 23], implemented on MapReduce framework, is divided into four steps. Data partitioning with  $\epsilon$ -replication is performed in the first step, local DBSCAN is performed in the second step, partition boundary points are identified for merging in the third step, and final merging is performed in the last step. However, the scalability of MR-DBSCAN [22, 23] is limited. There are a couple of GPU-based DBSCAN algorithms given in [1, 11].

Brecheisen et al. [9] gave a client-server model for parallel DBSCAN which uses OPTICS [2] for data partitioning to assign adjacent enumeration values to similar objects. Approximate clustering is obtained as lower-bounding distance values conservatively approximate the exact clustering. The clustering results of [34, 35] are comparable, but not identical to DBSCAN. Mr. Scan algorithm [35] was first introduced for GPGPU based systems. It is an approximate algorithm. It uses a tree-parallel approach for merging of local DBSCAN calculations. Authors have modified the DBSCAN algorithm to find the dense regions and infer their membership in a cluster without processing the points belonging to these dense regions. Merging is done on the basis of some representative points from each cluster resulting in an approximate solution. The same algorithm is implemented on a hybrid (CPU+GPGPU) system [34]. Recently, Patwary et al. gave two heuristic-based approximate algorithms, [29, 30] for DBSCAN. These algorithms use density-based sampling for efficient execution. They introduced two additional parameters to control the clustering quality and runtime. There is a tradeoff between clustering quality and execution time. Selection of these parameters depend on the data distribution and DBSCAN parameters. If these parameters are not chosen properly, the clustering quality of the algorithm may degrade and runtime may also increase. It is evident from the related work that the existing exact parallel algorithms are either very slow or do not scale well. The aim of this work is to give an efficient sequential DBSCAN and its scalable parallel solutions that reduce the neighborhood queries without compromising clustering quality.

GridDBSCAN uses a virtual grid over the data space to divide it into smaller size cells. The efficiency of GridDBSCAN is due to reduction in point-wise neighborhood queries for points in *dense enough* cells. There are two grid-based variants of DBSCAN [14, 36] available in literature. GRPDBSCAN [23] claims to reduce the sensitivity of DBSCAN on its parameters and better differentiates noise from clusters. GMDBSCAN [23] clusters high-dimensional data efficiently. The clustering results of these variants [14, 36] are different from classical DBSCAN. Very recently, a grid-based parallel implementations of DBSCAN, HPDBSCAN [18], has been proposed. In this approach, first equal size data is read by all  $p$  processors. Then, they assign data to overlay grid and redistribute it to  $p$  processors. It considers points in only neighboring cells of a cell  $C$  for efficient neighborhood query of points in  $C$ . The cost of querying all the points  $m$  in  $C$  is  $l*m$ , where  $l$  is total number of points in  $C$ 's neighborhood cells. The redistribution of data is performed using cost of cells i.e.  $totalcost/p$ , where  $totalcost$  is sum of  $l*m$  over all cells. This helps in achieving load balancing, but total cost will be much higher than that of classical DBSCAN in case of skewed data. This is because HPDBSCAN has query complexity linear in  $l$  in comparison to  $\log n$  in classical DBSCAN.

In this paper, we present a new parallel DBSCAN algorithm in which we parallelize a modified sequential DBSCAN algorithm, GridDBSCAN. The design of GridDBSCAN is such that it runs an order of magnitude faster than classical DBSCAN, enabling faster local computations, which constitute a major fraction of any parallel algorithm workload. The proposed parallel DBSCAN algorithm is shown to be scalable for all shared memory, distributed memory and hybrid memory systems.

### 3. DBSCAN

DBSCAN algorithm is capable of forming arbitrary shaped clusters for the given  $\mathcal{E}$  and  $MinPts$ . It computes  $\mathcal{E}$ -neighborhood for a point, and decides whether the point is a core point, a border point, or noise. A core point and its neighborhood form a cluster and the cluster is expanded by repeating this procedure on the neighboring points. Otherwise, the next point in the dataset is visited. A few useful definitions for DBSCAN are given as follows:

**$\mathcal{E}$ -neighborhood of a Point:** The  $\mathcal{E}$ -neighborhood of a point is a collection of points within its  $\mathcal{E}$ -radius.

**Core Point:** A core point with respect to  $\mathcal{E}$  and  $MinPts$  is a point that has at least  $MinPts$  points in its  $\mathcal{E}$ -neighborhood.

**Directly density-reachable (DDR):** A point  $p$  is directly density-reachable from a point  $q$  with respect to  $\mathcal{E}$  and  $MinPts$  if  $p$  is in  $\mathcal{E}$ -neighborhood of  $q$  and  $q$  is a core point.

**Density-Reachable (DR):** A point  $p$  is density-reachable from a point  $q$  with respect to  $\mathcal{E}$  and  $MinPts$  if there is a chain of points  $p_1, p_2, \dots, p_n, p_1 = q, p_n = p$  such that  $p_{i+1}$  is directly density reachable from  $p_i$  for all  $i$ .

**Density-Connected (DC):** A point  $p$  is density-connected to a point  $q$  with respect to  $\mathcal{E}$  and  $MinPts$  if there is a point  $o$  such that both  $p$  and  $q$  are density-reachable from  $o$ .

**Border Point:** A non-core point,  $p$  is a border point if it is density reachable from at least one of the core point.

**Noise:** A point is called noise if it is not directly density-reachable from any core point.

**Cluster:** A DBSCAN cluster is a maximal set of *density-connected* points.

The clustering obtained by classical DBSCAN is satisfying the following three conditions:

**Condition (Maximality):** For each pair  $p$  and  $q$ , if  $p \in S$  and  $q$  is density reachable from  $p$  with respect to  $\mathcal{E}$  and  $MinPts$ , then  $q \in S$  (where  $S$  is a cluster).

**Condition (Connectivity):** For each pair  $p, q \in S$ ,  $p$  is density connected to  $q$  with respect to  $\mathcal{E}$  and  $MinPts$ .

**Condition (Noise):** A point  $p$  belonging to the dataset is a noise point if  $p$  is not directly density reachable to any core point.

```

input: DataList X,  $\mathcal{E}$ , MinPts, union-find data structure UF
output: UF data structure containing clustering information
1: procedure DBSCAN (X,  $\mathcal{E}$ , MinPts)
2: for each point x  $\in$  X
3:   x.Nbh  $\leftarrow$  getNeighborhood(x,  $\mathcal{E}$ ) //x.Nbh is neighborhood of x
4:   if |x.Nbh|  $\geq$  MinPts
5:     mark x as a core point
6:     for q  $\in$  x.Nbh
7:       if q is a core point: union(UF, x, q)
8:       else if q does not belong to any cluster: union(UF, x, q)

```

**Figure 1. Disjoint-set based DBSCAN algorithm**

The detail pseudocode for disjoint-set based DBSCAN [27] implemented using *union-find* data structure is given in Figure 1. This algorithm constructs the cluster as a singleton tree for each point in the dataset and stores this information in *union-find* (UF) data structure. It repeatedly merges the clusters using the *union* (to

merge two tree's) and *find* (to find root node for a data point) operations. At the end of this process, each tree is a cluster. Singleton trees correspond to noise points.

The complexity of DBSCAN is  $n$  times the computation time for a region query i.e.,  $O(n \log n)$  (using a spatial indexing structure such as *R-tree* or its variant, *kd-tree*, etc.

### 4. THE PROPOSED GridDBSCAN

In this section, we propose a novel DBSCAN algorithm, GridDBSCAN that is fast but exact. The pseudocode for GridDBSCAN is given in Figure 2. GridDBSCAN performs virtual gridding on the dataset to exploit the spatial locality information to optimize the number and performance of neighborhood queries.

```

input: DataSet X
global:  $\mathcal{E}$ , MinPts, CellSize, Grid-R-Tree gT, union-find data structure UF
output: UF data structure containing clustering information
1: procedure GridDBSCAN(X)
2:   (gT, CL)  $\leftarrow$  buildGrid-R-Tree(X) //CL denotes cellList
3:   for each cell C  $\in$  CL: markCellType(C)
4:   for each C  $\in$  CL
5:     if C is CORE or DENSE: processCell(C)
6:     if C is SPARSE: processSparseCell(C)
7:   for each C  $\in$  CL
8:     if C is CORE or DENSE: mergeCluster(C)
9:   procedure markCellType(C)
10:    if |C|  $\geq$  MinPts
11:      mark C as CORE cell; C.hasCorePoint  $\leftarrow$  TRUE;
12:      for each x  $\in$  C: mark x as core point
13:    else: C.NbhList  $\leftarrow$  getNbhCells(C, gT)
        //returns list of cells in CellSize extended region
14:    if |C.NbhList|  $\geq$  MinPts
15:      mark C as DENSE cell; C.hasCorePoint  $\leftarrow$  TRUE;
16:      for each x  $\in$  C: mark x as core point
17:    else: mark C as SPARSE cell
18:   procedure processCell(C)
19:     C.NbhList  $\leftarrow$  getNbhdCells(C, gT)
20:     C.epsNbhList  $\leftarrow$  getEpsNbhdCells(C, gT)
        //C.epsNbhList does not include the cell which lies in C.NbhList
21:     for each C1  $\in$  C.NbhList
22:       if (C is CORE) OR ((C is DENSE) AND (C1 is CORE or DENSE))
23:         union_update(UF, C, C1) //replace C and C1 with their union in UF
24:       for each C1  $\in$  C.epsNbhList
25:         if (C1 is CORE or DENSE) AND (C.parent != C1.parent)
26:           add C1 to C.AuxList //C keeps a local cell list for post processing
27:   procedure processSparseCell(C)
28:     for each x  $\in$  C
29:       x.Nbh  $\leftarrow$  getNeighborhood(x, gT)
30:       if |x.Nbh|  $\geq$  MinPts
31:         for each q  $\in$  x.Nbh
32:           if (q is a core point) OR (q does not belong to any cluster)
33:             union_update(UF, x, q)
34:           else
35:             for each q  $\in$  x.Nbh
36:               if (q is a core point) AND (x does not belong to any cluster)
37:                 union_update(UF, x, q)
38:   procedure mergeCluster(C)
39:     for each C1  $\in$  C.AuxList
40:       if C.clusterID != C1.clusterID
41:         for each x  $\in$  C
42:           x.Nbh  $\leftarrow$  getNeighborhood(x, gT)
43:           for each q  $\in$  x.Nbh
44:             if (C.clusterID != q.clusterID) AND (q is a core point)
45:               union_update(UF, C, q)

```

**Figure 2. The proposed GridDBSCAN algorithm**

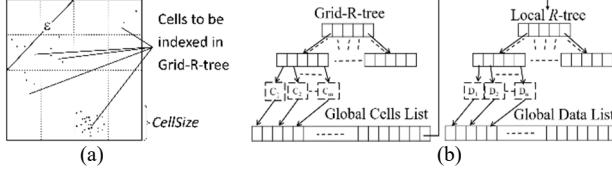
GridDBSCAN uses Grid-R-tree, a two- level variant of *R-tree* [21], where the top level, known as the global *R-tree*, stores cells and the

second level is called, local  $R$ -tree, constructed for each cell, containing the points in that cell. GridDBSCAN algorithm is divided into three major steps: 1) Grid- $R$ -tree construction, 2) determination of cell type, 3) processing of cells.

**Grid- $R$ -tree Construction:** Virtual gridding is achieved by splitting the data space into equal sized intervals in each dimension. This results in multi-dimensional hypercubes, which we refer to as cells (see Figure 3). The length of the interval is denoted by *cellSize*, computed by the equation:

$$\text{cellSize} = \mathcal{E}/(2\sqrt{d}), \quad (1)$$

where  $d$  is the number of dimensions. The chosen *cellSize* gives best performance while ensuring clustering results. The *cellSize* greater than this value does not guarantee exact clustering as that obtained by classical DBSCAN.



**Figure 3. (a) Gridding (b) Grid-R-tree data structure**

Grid- $R$ -tree is constructed by repeated insertion of points (see Figure 3). For each point, its containing cell is identified. If the cell is not already present, it is added to the global  $R$ -tree and to the global list of cells. The point is added to the local  $R$ -tree of the containing cell and is also added to the global list of points. These global lists enable efficient enumeration of cells and points.  $\mathcal{E}$ -neighborhood query of a point over Grid- $R$ -tree executes in a top-down recursive fashion from the root of global  $R$ -tree and successively reach to external node of the local  $R$ -tree.

To compute  $\mathcal{E}$ -neighborhood for a point,  $p$ , a region, *reg* is obtained by extending the coordinates of  $p$  by  $\mathcal{E}$  across all dimensions. The  $\mathcal{E}$ -neighborhood query executes in a top-down recursive fashion over the Grid- $R$ -tree, recursing itself into the children of a node whose MBRs overlap with the given *reg*. In its traversal, whenever it encounters an external node, it simply accumulates all the data points of this node that lie in region, *reg* into a temporary list and returns it.

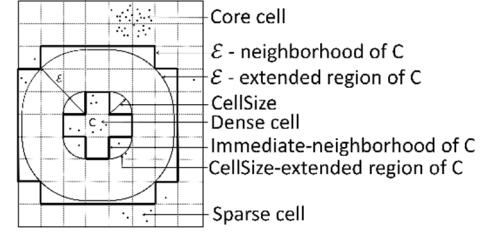
**Determination of Cell Type:** A cell is a core cell if it contains at least *MinPts* number of points. A cell is said to be *dense* if the number of points present in it and in its *immediate-neighborhood* cells is at least *MinPts*. It is said to be *sparse* otherwise (see Figure 4). Immediate-neighborhood of a cell includes those cells that are fully contained in its *cellSize*-extended region.

The global cell list is traversed to mark each cell as core, dense, or sparse. Immediate-neighborhood is computed from the Grid- $R$ -tree using a cell-wise neighborhood query on the *cellSize* extended region of a cell. The cell-wise neighborhood query is similar to neighborhood query in an  $R$ -tree, except that in this case the Global  $R$ -tree stores cells instead of points.

**Processing of Cells:** To process a core cell,  $C$ , its immediate-neighborhood cells are retrieved and are merged to  $C$  using the union-find data structure to expand the cluster initiated by  $C$ . This is possible because the size chosen for a cell (see Eq. 1) is such that all points contained in its immediate-neighborhood are within the  $\mathcal{E}$ -neighborhood of any point in  $C$ . All the core and dense cells that lie in  $C$ 's  $\mathcal{E}$ -neighborhood and do not belong to the same cluster, are inserted in an auxiliary list (*AuxList*) of  $C$ . The  $\mathcal{E}$ -neighborhood

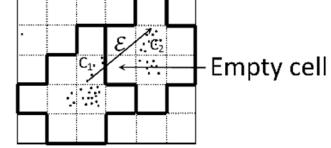
of a cell contains cells which lie, fully or partially, in the  $\mathcal{E}$ -extended region of the cell.

To process a dense cell,  $C$ , its immediate-neighborhood is retrieved. If an immediate neighboring cell is either dense or core, it is merged to  $C$ . All the core and dense cells that lie in  $C$ 's  $\mathcal{E}$ -neighborhood, and do not belong to the same cluster, are inserted in  $C$ 's *AuxList*. All core cells and dense cells are processed in a single scan. These are processed first in order to minimize point-wise merging subsequently.



**Figure 4. Cells and neighborhood for *MinPts* = 5**

To process a sparse cell,  $C$ , each point  $x$  in  $C$  must be processed. If  $x$  is a core point, then it is merged with the points in its  $\mathcal{E}$ -neighborhood. If  $x$  is a border point, then it is merged with a core point in its  $\mathcal{E}$ -neighborhood.



**Figure 5. Two core or dense cells not in immediate-neighborhood but in same cluster for *MinPts* = 5**

If two core or dense cells,  $C_1$  and  $C_2$ , that are not immediate neighbors of each other, it is possible that they were not considered for merging. But, they may belong to the same cluster, and they have to be considered for point-wise merging. It can be seen from Figure 5 that there can be an empty cell between  $C_1$  and  $C_2$ , and still they are in the same cluster. For this purpose, the *AuxList* of each cell is processed. For each cell  $C$  and a cell  $C'$  in *AuxList* of  $C$ , do:

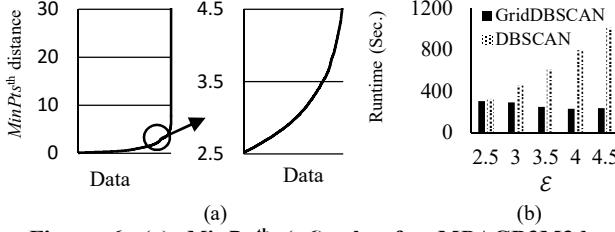
If  $C$  and  $C'$  belong to different clusters,  $\mathcal{E}$ -neighborhood of a point  $x$  in  $C$  is computed. If any point in  $C'$  falls within  $x$ 's  $\mathcal{E}$ -neighborhood then  $C$  and  $C'$  are merged; else the same process is repeated with the next point  $x'$  in  $C$ .

The likelihood of two such cells  $C$  and  $C'$  belonging to two different clusters would be very low because this would happen only if one or more empty cells occur among dense cells or core cells. If a sparse cell occurs among dense or core cells, it would have been already considered for point-wise merging.

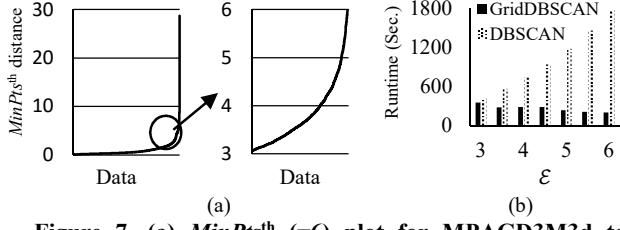
It has been seen empirically, point-wise neighborhood queries performed are much less.

It is observed that the runtime of GridDBSCAN decreases with increasing  $\mathcal{E}$ , because the number of cells is inversely proportional to  $\mathcal{E}$  (see Eq. 1) resulting in reduction in total number of neighborhood queries. In contrast, the runtime of classical DBSCAN increases with increasing  $\mathcal{E}$ , because the number of points in the  $\mathcal{E}$ -neighborhood is non-linearly proportional to  $\mathcal{E}$ . This is supported empirically in Figs. 6(b) and 7(b) (see Table 2 for dataset details). The value of  $\mathcal{E}$  affects the quality of clustering: a large  $\mathcal{E}$  may result in fewer clusters than that in natural clustering and the value of a small  $\mathcal{E}$  may result in more clusters than that in natural clustering.  $\mathcal{E}$  needs to be chosen carefully to ensure quality

clustering. We have empirically determined a range of  $\mathcal{E}$  values that produce quality clustering for a given dataset (see Figure 6(a) & 7(a)). For this, we have plotted  $MinPts^{th}$  nearest neighbor distance of points in the dataset (where  $MinPts=6$ ) in sorted order. The range of  $\mathcal{E}$  values are considered good for clustering, where there is a sharp change in distances [32] (2.5 to 4.5 for MPAGB3M3d and 3 to 6 for MPAGD3M3d). The curves at sharp change are zoomed and shown in the adjacent figures. We have verified that in the proposed range, GridDBSCAN performs much more efficiently than classical DBSCAN. For all the datasets used in experiments, GridDBSCAN outperforms classical DBSCAN by a factor of up to 8.98 (see Figure 7(b)).



**Figure 6. (a)**  $MinPts^{th}$  ( $=6$ ) plot for MPAGB3M3d to estimate  $\mathcal{E}$ . **(b)** Execution time



**Figure 7. (a)**  $MinPts^{th}$  ( $=6$ ) plot for MPAGD3M3d to estimate  $\mathcal{E}$ . **(b)** Execution time

## 4.1 Reduction in number of neighborhood queries

We now report the number of queries needed for GridDBSCAN for two datasets in Table 1 ( $q_1$ ,  $q_2$ , and  $q_3$  denote *cellSize*-neighborhood cell query,  $\mathcal{E}$ -neighborhood cell query, and  $\mathcal{E}$ -neighborhood point query, respectively). It can be observed from the table that the reduction in total number of queries required are 13% for MPAGD3M3d and 15% for MPAGB3M3d. For both the datasets,  $MinPts$  and  $\mathcal{E}$  are taken as 6 and 3, respectively. It can be noted that *CellSize*-neighborhood cell query takes lesser time than that of other queries because it is done only on global *R*-tree and *CellSize* is much smaller than  $\mathcal{E}$ . Therefore, actual speedup achieved is quite high than expected from reduction in number of neighborhood queries.

**Table 1. Neighborhood query analysis**

Data	$n$ (in M)	Total cells (in M)	$q_1$ (in M)	$q_2$ (in K)	$q_3$ (in K)	Query reduced (%)	Speedup achieved
MPAG B3M3d	3.37	1.11	1.6	600	661	15%	1.54
MPAG D3M3d	3.21	1.04	1.5	590	557	13%	1.20

## 4.2 Correctness of GridDBSCAN

We now prove that GridDBSCAN (as given in Figure 2) satisfies the three conditions: *maximality*, *connectivity*, and *noise* (see Sec. III). In the proof below  $x$  DDR  $y$ ,  $x$  DR  $y$ , and  $x$  DC  $y$  are used to denote the relations *directly-density-reachable*, *density-reachable*, and *density-connected* respectively between points  $x$  and  $y$ .

**Claim 1:** GridDBSCAN satisfies *Maximality* i.e. if  $x \in S$  and  $y$  DC  $x$  then GridDBSCAN ensures  $y \in S$ , where  $S$  is a cluster.

**Proof:**  $y$  DC  $x$  thus there is a common core point,  $p$  from which  $x$  and  $y$  are density reachable i.e. there is a series of points  $p=p_0, p_1, \dots, p_k=x$ , such that  $p_{i+1}$  DDR  $p_i$  and similarly  $p=q_0, q_1, \dots, q_l=y$  such that  $q_{l+1}$  DDR  $q_l$ .

Now it suffices to show that if  $y$  DDR  $q_{k-1}$  then  $y (=q_k)$  and  $q_{k-1}$  will be in the same cluster because then, by induction  $q_i$  and  $q_{i-1}$  are in the same cluster for all  $i$  and therefore  $y$  and  $p$  are in the same cluster. By symmetry  $p$  and  $x$  are in the same cluster i.e.  $x$  and  $y$  are in the same cluster.

We show in **Lemma 1** that  $y$  DDR  $q_{k-1} \rightarrow y (=q_k)$  and  $q_{k-1}$  will be in the same cluster. QED.

**Lemma 1:**  $y$  DDR  $q_{k-1} \rightarrow y \in \mathcal{E}$ -neighborhood of  $q_{k-1}$

Case  $q_{k-1}$  is in a core or dense cell:

- If  $y$  is in the immediate-neighborhood of  $q_{k-1}$ , both cells get merged to form a single cluster (line 21-23 in procedure *processCell*).
- If  $y$  is not in the immediate-neighborhood of  $q_{k-1}$  then if  $y$  is in a core/dense cell, then that cell is
  - either merged with  $q_{k-1}$ 's cell (Line 21-23 in procedure *processCell* ).
  - or put in AuxList of that cell (Line 24-26 in procedure *processCell* ). The cells in AuxList are considered for point-wise merging in procedure *mergeCluster*.
- If  $y$  is not in the immediate-neighborhood of  $q_{k-1}$  then if  $y$  is in a sparse cell:

Each point in a sparse cell (including  $y$ ) is considered for point-wise merger into a single cluster (*processSparseCell*)

Case  $q_{k-1}$  is in a sparse cell:

Each point in a sparse cell (including  $y$ ) is considered for point-wise merger into a single cluster (*processSparseCell*). QED.

**Claim 2:** GridDBSCAN satisfies *Connectivity* i.e. if GridDBSCAN adds points,  $p$  and  $q$  in the same cluster then  $p$  DC  $q$ .

**Proof:** GridDBSCAN adds a point  $x$  to a cluster with some core point  $y$  by merging (cells or point-wise) in Lines 23, 33, 37, and 45 in Figure 2. For each of these cases, we argue that  $p$  DC  $y$ .

Line 23 (procedure *processCell*): This procedure merges a core/dense cell  $C_1$  with cells  $C_1'$  in its immediate-neighborhood. For any points  $x$  in  $C_1$  and  $y$  in  $C_1'$ ,  $x$  DDR  $y$  because all cells in  $C_1$ 's immediate-neighborhood are completely contained in  $\mathcal{E}$ -neighborhood of  $x$  i.e.  $x$  DC  $y$ .

Line 33 (procedure *processSparseCell*): if  $x$  is a core point, then this line merges each point  $y$  in  $\mathcal{E}$ -neighborhood of  $x$  into the same cluster. Therefore  $y$  DDR  $x$  and so  $y$  DC  $x$ . [The same argument applies for a border point as well.]

Line 37 (procedure *processSparseCell*): If  $x$  is not a core point then this line merges  $x$  into a cluster containing a core point  $y$  in  $p$ 's  $\mathcal{E}$ -neighborhood. This implies  $x$  is in  $y$ 's  $\mathcal{E}$ -neighborhood and  $y$  is a core point i.e.  $x$  DDR  $y$  and so  $y$  DC  $p$ . The same argument applies for a border point as well.

Line 45 (procedure *mergeCluster*): Let  $C_1$  be a core /dense cell and  $x$  in  $C_1$  be a core point. If a point  $y$  in a cell  $C_1'$  in  $C_1$ 's AuxList is in a different cluster from that of  $x$ , it is added to the cluster of  $x$  but only if it is in the  $\mathcal{E}$ -neighborhood of  $x$ . Therefore  $y$  DDR  $x$  and so  $y$  DC  $x$ . QED.

**Claim 3:** GridDBSCAN satisfies condition *Noise* i.e. if  $x$  DDR  $y$  for some  $y$  then GridDBSCAN ensures  $x$  is in a cluster (i.e. it is not output as a noise point).

**Proof:** Let  $y$  be any core point. If  $x$  DDR  $y$ , then  $x$  will be in  $\mathcal{E}$ -neighborhood of  $y$ .

Case:  $y$  is in a core or a dense cell:

All cells in the  $\mathcal{E}$ -extended region of  $y$ 's cell will:

- either be merged with that cell (Line 23 in procedure *processCell*)
- or will be added to AuxList of that cell (Line 26 in procedure *processCell*) and merged later (Line 45 in procedure *mergeCluster*).

Either way,  $x$  will be in the same cluster as  $y$  i.e.  $x$  is not a noise point.

Case:  $y$  is in a sparse cell

A sparse cell is processed point-wise: since  $x$  is in  $\mathcal{E}$ -neighborhood of  $y$ , it will be added to the same cluster as  $y$ . i.e.  $x$  is not a noise point (Lines 33 and 37 in procedure *processSparseCell*). QED.

### 4.3 Complexity Analysis

The time complexity of GridDBSCAN, in the worst case, =  $O(n \log n)$ . In worst case GridDBSCAN behaves like DBSCAN. The worst case scenario occurs when all cells are sparse requiring neighborhood query to be performed for all data points. The worst case space complexity of GridDBSCAN is  $O(n)$ .

## 5. THE PROPOSED PARALLEL APPROACHES

GridDBSCAN is parallelized using the single program multiple data (SPMD) model. Parallel GridDBSCAN can be divided into three major steps: 1) data distribution, 2) local GridDBSCAN, and 3) merging local clustering results. For the first step, data is partitioned using *kd-tree* [16]. Each processing element runs GridDBSCAN independently on its local data in parallel in the second step. Finally, clusters obtained from local computations are merged to get DBSCAN clusters in a tree parallel mode. The pseudocode for parallel GridDBSCAN is given in Figure 9.

**Data Distribution:** Data partitioning strategy plays an important role in efficiency and scalability of a parallel algorithm. Preserving spatial locality of data points is critical for efficient DBSCAN execution. The *kd-tree* partitioning strategy, used in this paper, divides the data along dimensions such that each partition contains equal number of data points for load balancing and at the same time preserving spatial locality. We have replicated each partition's  $\mathcal{E}$ -extended strip to avoid communication overheads during local computations. The data partitioning and replication is illustrated in Figure 8 for 2-dimensional data. *kd-tree* typically works on binary splitting and thus restricts the number of partitions to the power of 2, but it can be generalized to work for any number of partitions. In this paper, we have experimented for  $p=2^k$  and  $p=3^k$ , where  $p$  is the number of partitions and  $k$  is a non-negative integer.

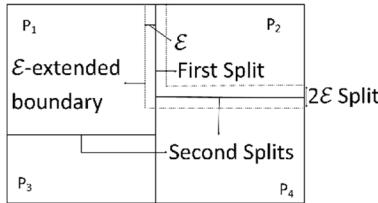


Figure 8. *kd-tree* data partitioning

**Local GridDBSCAN:** After receiving data in a balanced fashion, each processing element executes GridDBSCAN on local data. It constructs a local *Grid-R-tree* and cell list. Cell-wise computations are first performed on core and dense cells to merge most of the data points without point-wise processing. Fine-grained point-wise

processing is then performed over sparse cells and also on every cell,  $C$ , whose Auxlist contains at least one cell which has a different cluster membership than  $C$ .

```

input: DataList X, Number of processes p,  $\mathcal{E}$ , MinPts
output: A set of clusters in union-find data structure LL1
1: procedure parallel GridDBSCAN (X,  $\mathcal{E}$ , MinPts)
2:   for process P1
3:     buildKDTree(X, p); distribute data with  $\mathcal{E}$ -extended data
4:   for each process (proc) Pi 1 ≤ i ≤ p in parallel:
5:     GridDBSCAN (Xi,  $\mathcal{E}$ , MinPts) // Xi is local data at Pi
6:   for each proc Pi, in parallel:
7:     for (k = 1 to log2 p)
8:       if ((i-1)mod2k == 0)
9:         LL2 ← receive local LL1 from P(i+2k-1)
10:        LL1 ← mergeLL(LL1, LL2)
11:        // LL1, LL2 are local union-find at Pi and P(i+2k-1)
12:        if ((i-1)mod2k == 2k-1)
13:          send LL1 to P(i-2k-1) // LL1 received as LL2 on P(i-2k-1)
14: procedure mergeLL(LL1, LL2)
15:   for p ∈ LL1
16:     for q ∈ LL2
17:       if (p is a c.p & q is a c.p) or (p is a c.p & q is a b.p) or (p is a b.p & q is a c.p): union(LL1, p, q)
18:       else: add(LL1, q)
19:   return LL1

```

Figure 9. The proposed pGridDBSCAN algorithm

**Merging Local Clustering Results:** Local clustering results obtained from each processing element are merged in a tree parallel mode to get global clustering. Merging is very efficient since not a single neighborhood query is required in this step. For merging, we examine the  $2\mathcal{E}$  strip (see Figure 8) at the partition boundary to check for points which belong to two different clusters. These points are then used to merge clusters which span partitions. The steps involved in merging are given next:

The output of GridDBSCAN on a partition is a local list (*LL*) and the function, *MergeLL* (see Figure 9) traverses through two such lists, *LL*<sub>1</sub> and *LL*<sub>2</sub>, coming from two different processing elements. If a point  $p$  in *LL*<sub>1</sub> and a point  $q$  in *LL*<sub>2</sub> are the same, perform one of the following operations on that point:

- If  $p$  &  $q$  are core points in their respective partitions, merge the clusters containing them.
- If  $p$  is a core point in its partition and  $q$  is a border point, then merge the clusters containing them. Similar is the case when the roles of  $p$  and  $q$  are reversed.
- If both  $p$  and  $q$  are border points in their respective clusters, assign the point to any one of the two clusters.
- If  $p$  is a border point in its partition and  $q$  is a noise point, retain it as a border point.
- If both  $p$  and  $q$  are noise points in their respective partitions, retain it as a noise point.

Parallel GridDBSCAN satisfies the three conditions of maximality, connectivity, and noise of classical DBSCAN. We are omitting the proofs for lack of space.

**Complexity Analysis:** In our analysis, we are not factoring in the data distribution time as it is done offline. It is a common practice in literature. The time complexity of parallel GridDBSCAN (worst case) = local GridDBSCAN time + merging time =  $O((n/p) \log(n/p)) + O(n \log p)$  where,  $n$  is the total number of data points and  $p$  is the number of processing elements. Merging is done in a tree parallel mode taking  $\log p$  steps. The worst case space complexity of pGridDBSCAN is  $O(n)$ .

### 5.1 Distributed Memory GridDBSCAN

We have implemented parallel GridDBSCAN, called as dGridDBSCAN, for distributed memory systems. Each node has its own local memory and operates independently. dGridDBSCAN divides data into  $p$  partitions on a single node. It distributes the data including the data points in the respective  $\epsilon$ -extended strip. GridDBSCAN is performed in parallel on local data at each node. The merging of local clustering requires data communications. Each node communicates local union-find data structure via a message passing interface (MPI). Local clustering are merged in tree-parallel mode as explained in Figure 9. Our experimental results show that merging takes only a fraction of the total time taken by the algorithm. dGridDBSCAN is implemented using OpenMPI and is benchmarked against PDSDBSCAN-D.

### 5.2 Shared Memory GridDBSCAN

Parallel GridDBSCAN is implemented using OpenMP for shared-memory multi-core systems referred to as sGridDBSCAN. Since, locking is computationally expensive, each thread in sGridDBSCAN maintains its clustering information in a local union-find data structure, to avoid use of locks. Otherwise, locking is required to update the parent pointers of two points owned by two different threads. In this case, a thread has to wait for another thread when both are required to lock the same point at the same time. As a result, the wait time increases with increase in number of threads, limiting scalability.

sGridDBSCAN divides data into  $p$  partitions in the same way as that of dGridDBSCAN and assigns them to each thread. Each thread concurrently executes local GridDBSCAN. Local clustering results are then merged using a merging algorithm (see Figure 9) to get global clustering. However, it does not require any data communication like in dGridDBSCAN. sGridDBSCAN algorithm is benchmarked against PDSDBSCAN-S. Experimental results, given in Section 6, show that sGridDBSCAN has better scalability and efficiency than that of PDSDBSCAN-S.

### 5.3 Hybrid Memory GridDBSCAN

hGridDBSCAN is a hybrid version of GridDBSCAN for clusters having multicore nodes. It exploits multi-node parallelism of distributed memory system as well as multi-core parallelism on each node at the same time. Two levels of parallelization gives increased performance. Data distribution is done at two levels to take advantage of parallelism. First, data is divided into  $p$  partitions for  $p$  nodes of the cluster and then each node in turn divides its own data into  $t$  partitions for its  $t$  threads. Each thread of a node parallelly performs local GridDBSCAN. Similarly, merging is also performed at two levels, first at the thread level and then at the node level. The performance achieved by hGridDBSCAN is better than that of dGridDBSCAN (see Section 6). The time complexity of hGridDBSCAN (worst case) =  $O((n/pt) \log n/(pt)) + O(n \log p) + O(n \log t)$ .

## 6. EXPERIMENTAL SETUP AND RESULTS

For performance evaluation of proposed algorithms, we have considered various real datasets commonly used in literature for evaluating density-based clustering algorithms [19, 20, 27, 30]. DGalaxiesBower2006a (DGB) [8], MPAGalaxiesBertone2007a (MPAGB) [7], MPAGalaxiesDelucia2006a (MPAGD) [25], FOF [31], and MPAHaloTreesMhalo (MPAH) [7] datasets are obtained from the Galaxy and Halo databases on Millennium Run. In addition, we have also used a few other real datasets: DGalaxiesFont2008a (DGF) [25], 3D Road Network (3DRN) [24]

and household Power Consumption (HHP) [33]. Details of structural properties of the datasets and input parameters are given in Table 2. We have used  $kd$ -tree (see Section 5) for geometrically partition the data among the processing elements. However, we do not include data partitioning and file read/write operations while computing the speedup of parallel GridDBSCAN.

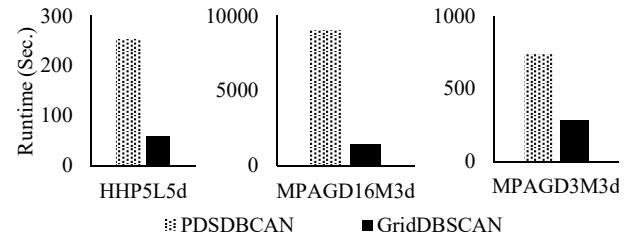
For each dataset, the execution times of GridDBSCAN has been benchmarked against sequential PDSDBCAN [27]. sGridDBSCAN and dGridDBSCAN are benchmarked against PDSDBCAN-S [27] and PDSDBCAN-D [27], respectively. The speedup of the proposed parallel algorithms over GridDBSCAN is also reported. Some time measurements are taken using VampirTrace version 5.14.2 [10] and some are taken using MPI\_Wtime. Grid-R-tree parameters  $m=4$  (min-entries) and  $M=8$  (max-entries) have been used for all experiments.

**Table 2. Dataset specification**

Dataset	$n$	$d$	$\epsilon$	$MinPts$	Cluster	Noise
3DRN	434K	3	0.01	6	316	2554
DGB8M3d	8.7M	3	2	6	341	71
MPAGB3M3d	3.4M	3	4	6	241	3141
MPAGB8M3d	8M	3	2	6	10466	108234
MPAH1M9d	919K	9	30	6	525	45750
MPAH1.5M9d	1.5M	9	30	6	2619	79511
MPAGD3M3d	3.2M	3	4	6	483	4053
MPAGD8M3d	8M	3	2	6	9416	102963
MPAGD16M3d	16M	3	2	6	9583	102919
HHP5L5d	475K	5	0.5	6	141	13437
FOF11M3d	11.4M	3	1	6	521712	2610848
FOF28M3d	28.5M	3	1	6	1035490	1891990
FOF57M3d	57M	3	1	6	1613820	0

### 6.1 Results for GridDBSCAN

In Section 4, we have claimed that GridDBSCAN is efficient than any existing sequential DBSCAN. In support of this, we compare the runtime of GridDBSCAN with the state-of-the-art sequential PDSDBSCAN [27] for few real datasets. From Figure 10, it is clear that GridDBSCAN is significantly faster than PDSDBSCAN.



**Figure 10. Runtime comparison between GridDBSCAN and PDSDBSCAN**

### 6.2 Results for dGridDBSCAN

To perform the experiments for dGridDBSCAN, we use two different distributed memory systems – 1) 32 nodes Beowulf cluster. Each node having single quad core processors (3.00GHz Intel(R) Xeon(R) 5160) and 16 GB RAM. There is a separate master node for cluster management and it is not used for running the code. The nodes are connected by 48-port, Gigabit Ethernet switches. No local I/O is used. Input data is fetched from a NAS server on the local network and the output file is written onto the same NAS server. 2) An advanced Supercomputing Hub for OMICS Knowledge in Agriculture (ASHOKA) [5] consists of 256 nodes Linux cluster with two masters. We experimented up to 128 nodes of distributed memory server (hn1) of ASHOKA.

The code for this set of experiments is written using OpenMPI [17] and C. All time measurements on ASHOKA are taken using MPI\_Wtime function. dGridDBSCAN uses a single thread at every node. We have used various real datasets for performance evaluation of our distributed implementation to show the scalability for increasing number of nodes.

Figs. 11(a) & 11(b) show the speedup achieved by dGridDBSCAN over GridDBSCAN on hn1 server of ASHOKA. The speed up achieved on 128 nodes for different datasets with size varying from 1 million data points (with 9 dimensions) to 18 million data points (with 3 dimensions) is in the range 63 to 178 when only local computations are considered. When merging time is also factored in, the speed up achieved ranges from 61 to 152.

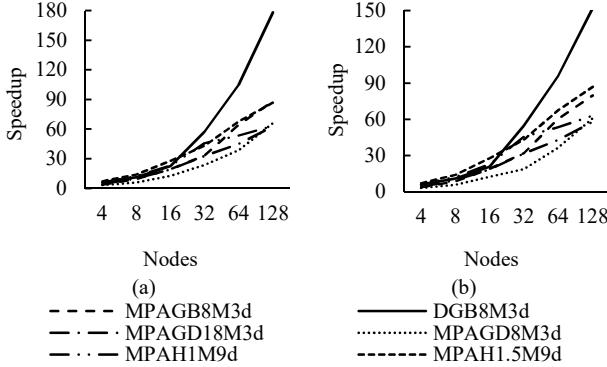


Figure 11. Speedup for (a) local GridDBSCAN (b) local GridDBSCAN and merge time

In Figs. 12 (a) & 12 (b), we show the scalability of the algorithm with increasing number of data points (11 million to 56 million). In Figs. 11 & 12, super linear speed up can be attributed to the reduced query time in Grid-R-tree for partitioned data.

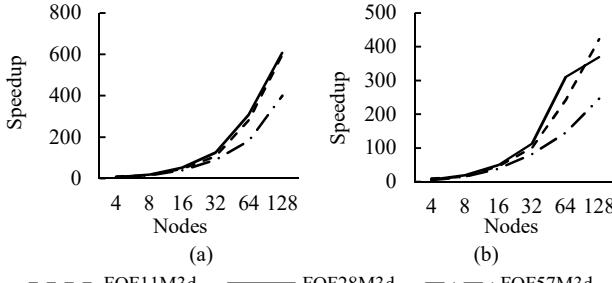


Figure 12. Speedup for (a) local GridDBSCAN (b) local GridDBSCAN and merge time

Table 3 shows the query analysis of FOF3M3d for achieving super-linear speedup in dGridDBSCAN at  $p=32$ .  $t_1$ ,  $t_2$ , and  $t_3$  denote per query speedup for  $\text{cellSize}$ -neighborhood cell query,  $\mathcal{E}$ -neighborhood cell query, and  $\mathcal{E}$ -neighborhood point query, respectively. The effective speedup that the algorithm can achieve is approximately 278.

Table 3. Neighborhood query analysis

Data	$t_1$	$t_2$	$t_3$	Actual speedup achieved
FOF3M3d	9.69	8.23	8.19	113.57

Figure 13 compares dGridDBSCAN with PDSDBSCAN-D for local and local+merge times for two real datasets on 32 nodes. It is clear from the results that dGridDBSCAN's running time is considerably less than PDSDBSCAN-D and the time taken for local computations and merging is considerably faster. We could

not conduct these experiments on more than 32 nodes because of unavailability of hn1 server of ASHOKA [5].

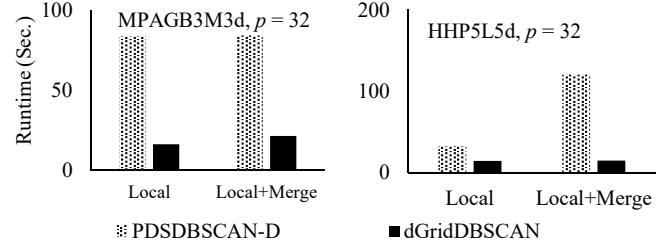


Figure 13. PDSDBSCAN-D and dGridDBSCAN execution time comparison

Figure 14 shows the time taken by various constituents of dGridDBSCAN and their behavior with increasing number of nodes. We can observe that time for Grid-R-tree construction and local computation is decreasing with increasing number of nodes. Merging time is increasing for increasing number of nodes, but the rate of increase is not significant and merging time is a small fraction of total time.

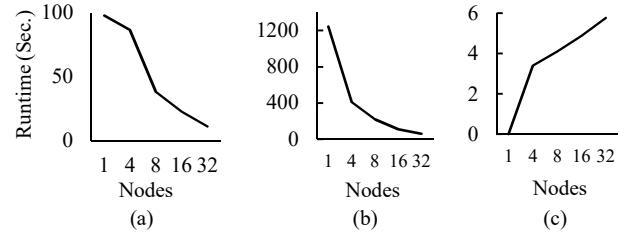


Figure 14. (a) Grid-R-tree construction (b) Local computation (c) Merging time for MPAGB8M

### 6.3 Results for sGridDBSCAN

All shared memory experiments have been carried out on a HP ProLiant DL 580 G8 server with four processors and 200 GB of global RAM. Each processor is an Intel (R) Xeon (R) CPU E7-4850 v2 @ 2.30 GHz with twelve cores. In all, there are 48 cores capable of supporting 96 threads. The operating system used is CentOS Linux 6.5. The code is implemented using OpenMP [26] and C.

We have benchmarked sGridDBSCAN with PDSDBSCAN-S [27], which is the best existing shared memory implementation of DBSCAN. Figure 15 shows the execution time for both the algorithms and it is found that sGridDBSCAN is significantly faster than PDSDBSCAN-S. The improved performance of sGridDBSCAN can be attributed to reduction in the number of point-wise neighbourhood queries and to the way cell queries are optimized.

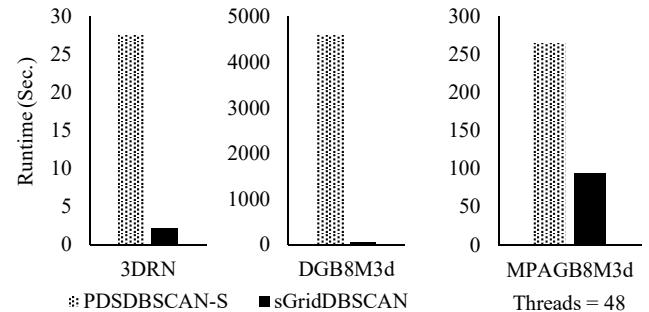
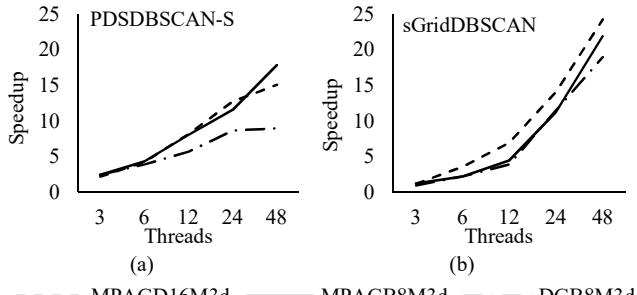


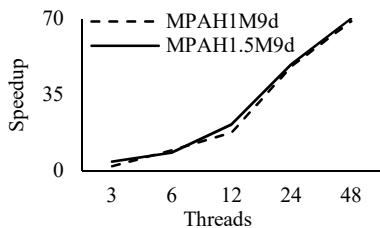
Figure 15. Execution time comparison of sGridDBSCAN and PDSDBSCAN-S

In Figure 16, we compare the speedup of sGridDBSCAN and PDSDBSCAN-S and it is clear that sGridDBSCAN achieves better speedup and scalability than the best existing shared memory parallel algorithm.



**Figure 16. Scalability comparison of (a) PDSDBSCAN-S and (b) sGridDBSCAN for varying threads**

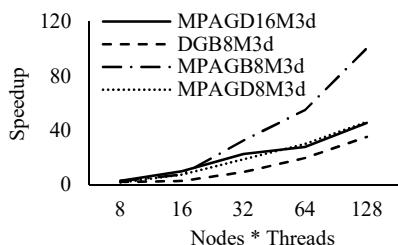
Figs. 16(b) & 17 show speedup obtained for sGridDBSCAN with increasing number of threads with respect to GridDBSCAN (executed on a single thread) for different datasets having varying size and dimensions. In all cases, the speedup obtained is up to 67.



**Figure 17. Speedup of sGridDBSCAN**

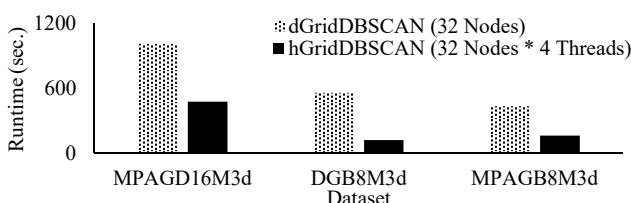
#### 6.4 Results for hGridDBSCAN

Experiments for hGridDBSCAN are conducted on the 32-node Beowulf cluster. Figure 18 shows speedup for various datasets. The speedup is computed on  $p$  nodes by exploiting 4 threads on each node with respect to sequential GridDBSCAN (only one thread). The maximum speedup achieved is 100.63 using 128 (32 nodes \* 4 threads) threads.



**Figure 18. Speedup comparison of hGridDBSCAN**

In Figure 19, the execution time of hGridDBSCAN is compared with dGridDBSCAN using 4 datasets on a 32-node quad-core cluster. The efficiency of hGridDBSCAN is found to be reasonably good.



**Figure 19. Comparison of hGridDBSCAN with dGridDBSCAN**

Experimental results demonstrate that our shared, distributed, and hybrid memory algorithms perform consistently better than respective best existing shared memory (PDSDBCAN-S [27]) and distributed memory (PDSDBCAN-D [27]) algorithms on all datasets considered in this paper. This is because, GridDBSCAN reduces number of queries and in addition to that executes queries efficiently in comparison to the number of point queries required in existing algorithms. Moreover, our efficient tree parallel mode of merging has minimal communication overhead.

## 7. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a novel sequential DBSCAN algorithm which outperforms the state-of-the-art sequential DBSCAN by exploiting spatial information and reduced/optimized neighborhood queries. A parallel GridDBSCAN algorithm is also proposed which is implemented on shared memory, distributed memory and on multicore cluster systems. The proposed parallel algorithm scales well with increasing data-size and processing elements, due to efficient and effective parallelization strategy. This is observed across all parallel architectures and datasets considered for experiments. The maximum speedup achieved for shared, distributed, and hybrid models are 66.99 on 48 core machine, 423.1 on 128 nodes cluster, and 110.57 on 32 nodes cluster each having 4 threads respectively. The parallel implementations are benchmarked against the best existing parallel solutions. Scalability and speedups are achieved without compromising the quality of clustering i.e. clustering results of proposed sequential and parallel algorithms are identical to the state-of-the-art DBSCAN.

We next plan to work on a data distribution layer for intelligently distributing data among processing elements. This would further increase the efficiency of our parallel algorithm to scale for much larger datasets and also for increasing number of processing elements.

## 8. ACKNOWLEDGMENTS

This work was supported by a Research Grant from Department of Electronics and Information Technology, Government of India.

## 9. REFERENCES

- [1] Andrade, G., Ramos, G., Madeira, D., Sachetto, R., Ferreira, R. and Rocha, L. 2013. G-DBSCAN: A {GPU} Accelerated Algorithm for Density-based Clustering. *Procedia Computer Science*. 18, (2013), 369–378.
- [2] Ankerst, M., Breunig, M.M., Kriegel, H. and Sander, J. 1999. OPTICS: Ordering Points To Identify the Clustering Structure. *Proc. ACM SIGMOD'99 Int. Conf. on Management of Data* (1999), 49–60.
- [3] Aoying, Z., Shuigeng, Z., Jing, C., Ye, F. and Yunfa, H. 2000. Approaches for scaling DBSCAN algorithm to large spatial databases. *J. Comput. Sci. Technol.* 15, 6 (2000), 509–526.
- [4] Arlia, D. 2001. Experiments in Parallel Clustering with DBSCAN. in the *Proc. of the Euro Par '01 conference, LNCS* (2001), 326–331.
- [5] ASHOKA- Advanced Supercomputing Hub for OMICS Knowledge in Agriculture: 2014. <http://webapp.cabgrid.res.in:8086/pbsworks/ui/framework/login/login.html>.
- [6] Beckmann, N., Kriegel, H.-P., Schneider, R. and Seeger, B. 1990. The R\*-tree: An Efficient and Robust Access Method for Points and Rectangles. *Proc. of the 1990 ACM SIGMOD Int. Conf. on Management of Data* (1990), 322–331.
- [7] Bertone, S., De Lucia, G. and Thomas, P.A. 2007. The recycling of gas and metals in galaxy formation: Predictions

- of a dynamical feedback model. *Mon. Not. Roy. Astron. Soc.* 379, (2007), 1143–1154.
- [8] Bower, R.G., Benson, A.J., Malbon, R., Helly, J.C., Frenk, C.S., Baugh, C.M., Cole, S. and Lacey, C.G. 2006. Breaking the hierarchy of galaxy formation. *Monthly Notices of the Royal Astronomical Society*. 370, 2 (2006), 645–655.
- [9] Brecheisen, S., Kriegel, H. and Pfeifle, M. 2006. Parallel Density-Based Clustering of Complex Objects. *Advances in Knowledge Discovery and Data Mining*. (2006), 179–188.
- [10] Brunst, H., Hackenberg, D., Juckeland, G. and Rohling, H. 2009. Comprehensive Performance Tracking with Vampir 7. *Tools for High Performance Computing 2009*. Springer Berlin Heidelberg. 17–29.
- [11] Chen, C.-C. and Chen, M.-S. 2015. HiClus: Highly Scalable Density-based Clustering with Heterogeneous Cloud. *Procedia Computer Science*. 53, (2015), 149–157.
- [12] Chen, M., Gao, X. and Li, H. 2010. Parallel DBSCAN with Priority R-tree. *Information Management and Engineering (ICIME), 2010 The 2nd IEEE Int. Conf. on* (2010), 508–511.
- [13] Coppola, M. and Vanneschi, M. 2002. High-performance data mining with skeleton-based structured parallel programming. *Parallel Computing*. 28, 5 (2002), 793–813.
- [14] Darong, H. and Peng, W. 2012. Grid-based DBSCAN Algorithm with Referential Parameters. *Physics Procedia*. 24, Part B, (2012), 1166–1170.
- [15] Ester, M., Kriegel, H., S, J. and Xu, X. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining* (1996), 226–231.
- [16] Di Fatta, G. and Pettinger, D. 2010. Dynamic Load Balancing in Parallel KD-Tree k-Means. *2010 10th IEEE Int. Conf. on Computer and Information Technology*. Cit (2010), 2478–2485.
- [17] Gabriel, E., Fagg, G.E., Bosilca, G., Angskun, T., Dongarra, J.J., Squyres, J.M., Sahay, V., Kambadur, P., Barrett, B., Lumsdaine, A., Castain, R.H., Daniel, D.J., Graham, R.L. and Woodall, T.S. 2004. Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation. *Proc., 11th European PVM/MPI Users' Group Meeting* (Budapest, Hungary, 2004), 97–104.
- [18] Götz, M., Bodenstein, C. and Riedel, M. 2015. HPDBSCAN: Highly Parallel DBSCAN. *Proc. of the Workshop on Machine Learning in High-Performance Computing Environments* (NY, USA, 2015), 2:1–2:10.
- [19] Goyal, P., Kumari, S., D, K., Balasubramaniam, S. and Goyal, N. 2014. Parallelizing OPTICS for Multicore Systems. *Proc. of the 7th ACM India Computing Conf.* (2014), 17:1–17:6.
- [20] Goyal, P., Kumari, S., Kumar, D., Balasubramaniam, S., Goyal, N., Challa, J.S. and Islam, S. 2015. Parallelizing OPTICS for Commodity Clusters. *Proc. of the 2015 Int. Conf. on Distributed Computing and Networking- ICDCN'15* (NY, USA, 2015), 1–10.
- [21] Guttman, A. 1984. R-Trees. a Dynamic Index Structure for Spatial Searching. *Proc. of the 1984 ACM SIGMOD Int. Conf. on Management of Data* (1984), 47–57.
- [22] He, Y., Tan, H., Luo, W., Feng, S. and Fan, J. 2014. MR-DBSCAN: A scalable MapReduce-based DBSCAN algorithm for heavily skewed data. *Frontiers of Computer Science*. 8, 1 (2014), 83–99.
- [23] He, Y., Tan, H., Luo, W., Mao, H., Ma, D., Feng, S. and Fan, J. 2011. MR-DBSCAN: An Efficient Parallel Density-Based Clustering Algorithm Using MapReduce. *Proc. of the 2011 IEEE 17th Int. Conf. on Parallel and Distributed Systems* (Washington, DC, USA, 2011), 473–480.
- [24] Kaul, M., 0002, B.Y. and Jensen, C.S. 2013. Building Accurate 3D Spatial Networks to Enable Next Generation Intelligent Transportation Systems. *2013 IEEE 14th Int. Conf. on Mobile Data Management* (Milan, 2013), 137–146.
- [25] De Lucia, G. and Blaizot, J. 2007. The hierarchical formation of the brightest cluster galaxies. *Mon. Not. Roy. Astron. Soc.* 375, (2007), 2–14.
- [26] Mohr, B., Malony, A.D., Shende, S. and Wolf, F. 2002. Design and prototype of a performance tool interface for OpenMP. *Journal of Supercomputing*. 23, 1 (2002), 105–128.
- [27] Patwary, M.A., Palsetia, D., Agrawal, A., Liao, W., Manne, F. and Choudhary, A. 2012. A new scalable parallel DBSCAN algorithm using the disjoint-set data structure. *Proc. of the Int. Conf. on High Performance Computing, Networking, Storage and Analysis* (LA, CA, USA, 2012), 62:1–62:11.
- [28] Patwary, M.M.A., Blair, J. and Manne, F. 2010. Experiments on Union-find Algorithms for the Disjoint-set Data Structure. *Proc. of the 9th Int. Conf. on Experimental Algorithms* (Berlin, Heidelberg, 2010), 411–423.
- [29] Patwary, M.M.A., Byna, S., Satish, N.R., Sundaram, N., Lukic, Z., Roytershteyn, V., Anderson, M.J., Yao, Y., Prabhat and Dubey, P. 2015. BD-CATS: big data clustering at trillion particle scale. *SC* (2015), 6:1–6:12.
- [30] Patwary, M.M.A., Satish, N., Sundaram, N., Manne, F., Habib, S. and Dubey, P. 2014. Pardicle: Parallel Approximate Density-based Clustering. *Proc. of the Int. Conf. for High Performance Computing, Networking, Storage and Analysis* (Piscataway, NJ, USA, 2014), 560–571.
- [31] Springel, V. et al. 2005. Simulations of the formation, evolution and clustering of galaxies and quasars. *Nature*. 435, 7042 (2005), 629–636.
- [32] Tan, P.-N., Steinbach, M. and Kumar, V. 2005. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc.
- [33] UCI Machine Learning Repository: 2013. <http://archive.ics.uci.edu/ml>.
- [34] Welton, B. and Miller, B.P. 2014. Mr. Scan: A Hybrid/Hybrid Extreme Scale Density Based Clustering Algorithm. (2014).
- [35] Welton, B., Samanas, E. and Miller, B.P. 2013. Mr. Scan: Extreme scale density-based clustering using a tree-based network of GPGPU nodes. *High Performance Computing, Networking, Storage and Analysis (SC), 2013 Int. Conf. for* (2013), 1–11.
- [36] Xiaoyun, C., Yufang, M., Yan, Z. and Ping, W. 2008. GMDBSCAN: Multi-Density DBSCAN Cluster Based on Grid. *e-Business Engineering, 2008. ICEBE '08. IEEE Int. Conf. on* (2008), 780–783.
- [37] Xu, X., Jäger, J. and Kriegel, H.-P. 1999. A Fast Parallel Clustering Algorithm for Large Spatial Databases. *Data Mining and Knowledge Discovery*. 3, 3 (1999), 263–290.
- [38] Yan Xiang Fu Wei Zhong Zhao, H.F.M., Zhao, W.Z., Ma, H.F. and Fu, Y.X. 2011. Research on Parallel DBSCAN Algorithm Design Based on MapReduce. *Advanced Measurement and Test* (2011), 1133–1138.
- [39] Goyal, P., Challa, J.S., Kumar, D., Balasubramaniam, S., and Goyal, N. 2016. Grid-R-tree: A data structure for efficient neighborhood and nearest neighbor queries in data mining. *Data & Knowledge Engineering*, Elsevier (Under Review).

# Scalable MapReduce-based Fuzzy Min-Max Neural Network for Pattern Classification

Shashikant Ilager

School of Computer and Information Sciences  
University of Hyderabad  
Hyderabad  
Telangana, India  
shashikant.ilager@gmail.com

Dr. P.S.V.S Sai Prasad

School of Computer and Information Sciences  
University of Hyderabad  
Hyderabad  
Telangana, India  
saics@uohyd.ernet.in

## ABSTRACT

Fuzzy Min-Max Neural Network (FMNN) is a pattern classification algorithm which incorporates fuzzy sets and neural network. It is most suitable for online algorithms. Based on this, a MapReduce-based Fuzzy Min-Max Neural Network (MRFMNN) algorithm for pattern classification is proposed using Twister framework. MapReduce approach is used for scaling up the FMNN for massive large scale datasets. We used standard membership, expansion and the contraction functions of the traditional FMNN algorithm. The performance of the MRFMNN is tested by using several benchmark and synthetic datasets against the traditional FMNN. Results empirically established that MRFMNN achieves significant computational gains over FMNN without compromising classification accuracy.

## Keywords

Neural Network, Fuzzy Sets, Classification, FMNN, MapReduce, Twister, MRFMNN

## 1. INTRODUCTION

The introduction of the MapReduce programming model [3], has changed the scenario of distributed computing by giving a framework for the development of scalable algorithms for big data problems. Converting the existing traditional algorithms to the MapReduce approach is the need of the hour to deliver quick results on large scale datasets. MapReduce framework incorporates data parallelism and utilizes the underlying distributed heterogeneous resources effectively. Several frameworks are designed to implement the MapReduce programming model. Google's Hadoop [3], Apache Hadoop [2], Apache Spark [14], [1] and Twister [4] are some of those. In this work, Twister is used to implement the FMNN for MapReduce approach.

The strength of the neural network, which is capable of simulating computers to work like a human brain and nervous system, has led to several innovative solutions in the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

*ICDCN '17, January 04-07, 2017, Hyderabad, India*

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007776>

field of artificial intelligence. After the introduction of Fuzzy sets [13], several computing models and algorithms have been proposed based on Fuzzy logic. The combination of fuzzy logic and the artificial neural network has resulted in hybrid intelligent system popularly known as Fuzzy Neural Network or Neuro-Fuzzy systems [11]. The combination of these systems makes use of the capability of each other and reduces their individual shortcomings. Fuzzy logic also helps neural networks to converge faster.

Fuzzy Min-Max Neural Network (FMNN) [10] was proposed by Simpson for pattern classification. It is a nonlinear, separable, online adaptive and one pass learning pattern classifier. MapReduce framework is more efficient for non-iterative algorithms. As FMNN is a single pass algorithm, a single MapReduce job is sufficient for constructing the classifier. Hence, FMNN is the best candidate for the construction of MapReduce-based classifier. Several improvements are proposed to the basic FMNN, and many refined solutions are given. Compensatory neuron based FMNN [9], center gravity data based FMNN [7], data core based FMNN [15] and enhanced FMNN [8] are some of those. All these different versions are suggested several changes to the basic FMNN to improve the performance and accuracy. In this work, we considered basic FMNN for conversion to the MapReduce approach.

The remaining paper is organized as follows: Section 2 gives an overview of fuzzy sets, traditional FMNN, and Twister. Section 3 describes proposed MapReduce approach to the FMNN. Results and analysis carried out are shown in Section 4. Finally, conclusions are drawn in Section 5.

## 2. BACKGROUND

### 2.1 Fuzzy Set

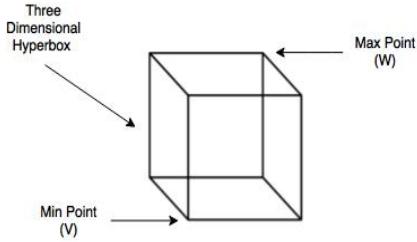
Theory of fuzzy sets was given by Zadeh [13]. Fuzzy sets represent the objects for the real-time nature where object precisely do not belong to a particular class. The membership function is proposed, which calculates membership value for all the objects. Unlike in traditional sets where objects are classified in a crisp manner where membership of an object to corresponding class is either 0 or 1, in fuzzy sets, objects have a partial membership for corresponding class ranging from [0,1]. The degree or probability of an object belonging to a particular class depends on the membership value. The membership value near to 1 indicates the object is more likely to belong to that class and 0 indicates that the object does not belong to the respective class.

## 2.2 Traditional FMNN

Simpson introduced pattern classification algorithm FMNN by using hyperbox fuzzy sets and neural networks [10]. It is a three layered neural network. FMNN algorithm is used for the construction of the classifier on datasets consisting of the numerical conditional attributes and categorical decision attribute.

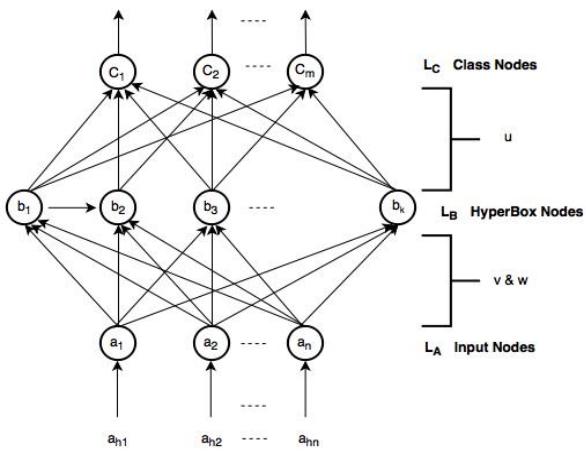
Let  $C_1, C_2 \dots C_m$  denote the decision classes and let  $n$  be the number of conditional attributes. As a preprocessing step, all attributes are normalized to the unit range  $(0, 1)$ , hence the entire space of the objects becomes an  $n$  dimensional unit cube  $I^n$ .

In the process of learning, hyperboxes are constructed such that all objects in a hyperbox belong to a unique decision class. A Hyperbox is a  $n$  dimensional cube represented by two  $n$  dimensional points  $V$  and  $W$  denoting the minimum corner and the maximum corner as shown in the Figure 1.



**Figure 1: A Three Dimensional Hyperbox with Minimum Point V and Maximum Point W**

The resulting FMNN, comprising the hyperboxes is represented as a three layer neural network as depicted in the Figure 2.



**Figure 2: Traditional FMNN- A Three Layer Neural Network**

The  $n$  dimensional input layer  $L_A$  corresponds to receiving the data for one training object with  $n$  conditional attributes. A node in a Hidden layer layer  $L_B$  represents an hyperbox and  $L_A$  and  $L_B$  are fully connected. The connecting edge from  $i^{\text{th}}$  input node to the  $j^{\text{th}}$  hidden layer node is associated with the two weight values  $v_{ij}, w_{ij}$  denoting  $i^{\text{th}}$

component of minimum ( $V$ ) and maximum ( $W$ ) points of the hyperbox  $b_j$ .

The output layer is of the size  $m$  corresponding to the  $m$  decision classes. As each hyperbox belongs to a single decision class, the weight vector  $u$  coming out of a hyperbox in a hidden layer is a boolean vector having all zeroes except in for the connection to its decision class.

The weights between the input layer to the hidden layer are represented by minimum point matrix ( $v$ ) of size  $n/k$  and maximum point matrix ( $w$ ) of size  $n/k$ . The boolean vector matrix between hidden layer and the output layer is represented by  $u$  of the size  $k/n$ , where  $k$  is the total number of hyperboxes constructed at the hidden layer.

### 2.2.1 FMNN Training Algorithm

FMNN neural network training comprises three steps. From the training dataset, each instance or object is passed to the hidden layer. The membership value of an object is calculated using the membership function [10]. Let  $A_h = (a_{h1}, a_{h2}, \dots, a_{hn}) \in I^n$  is the  $h^{\text{th}}$  input training object,  $V_j = (v_{j1}, v_{j2}, \dots, v_{jn})$  is the minimum point for hyperbox  $B_j$  and  $W_j = (w_{j1}, w_{j2}, \dots, w_{jn})$  is the maximum point of the  $B_j$ .  $\gamma$  acts as sensitivity parameter that controls how fast membership value decrease as the distance between  $A_h$  and  $B_j$  increases. Let  $b_j(A_h)$  is membership function for the  $j^{\text{th}}$  hyperbox  $0 \leq b_j(A_h) \leq 1$ , it is defined as the following equation 1.

$$b_j(A_h) = \frac{1}{2n} \sum_{i=1}^n [\max(0, 1 - \max(0, \gamma \min(1, a_{hi} - w_{ji}))) + \max(0, 1 - \max(0, \gamma \min(1, v_{ji} - a_{hi})))] \quad (1)$$

Initially, in case if no hyperbox of the corresponding class exists, a new point hyperbox is created with the same dimension as that of the current instance of the training object ( $V_j = W_j = A_h$ ). Otherwise, fuzzy membership value is calculated for all the hyperboxes belonging to the same class.

For every training object, membership values are calculated for the corresponding hyperboxes (hyperboxes which belong to the same decision class). The training object is included to the hyperbox to which it is fully contained (membership value = 1). If the training object is not fully contained in any of the hyperbox, then the nearest hyperbox is identified which has the highest membership value. The expansion criteria are checked for the identified hyperbox. If the expansion criteria are satisfied, the hyperbox is expanded to contain the training object. If the expansion criteria are not satisfied, a new point hyperbox is included in the hidden layer such that  $V_j = W_j = A_h$ . After every expansion, overlap test is carried out to identify the overlapped region between two hyperboxes belonging to two different classes. If overlap exists, the contraction is done to remove the overlapped region. The detailed equations for expansion test, overlap test, and contraction process are present in [10].

### 2.2.2 FMNN Testing Algorithm

In FMNN testing algorithm, the test case or test object is passed to the all the neurons (hyperboxes) at the hidden layer. Fuzzy membership value is calculated for each hyperbox. At the output layer, the highest membership value

instance is chosen among multiple membership values. The corresponding class of the hyperbox is the class of the test case.

### 2.3 Twister: An Iterative MapReduce Framework

MapReduce is a programming model which facilitates to deal with the large size data. Twister[4] is a light-weight framework which implements MapReduce programming model. Unlike popular framework like Hadoop [12], Twister also supports in-memory computation for the iterative MapReduce algorithms. It effectively keeps all the initial loaded data in the main memory until all the iterations are over, hence reducing the cost of the I/O and computation time.

Twister framework has components like Driver, Mapper, Reducer, Collector, and Combiner. The driver acts as the main program. Twister categorizes the data into two types. First, the static data, which was loaded initially and kept in memory until all iterations are over. Second, dynamic data that changes over iteration. The static data which is huge in size is partitioned into several parts and kept in distributed locations in the cluster. All mappers load associated static data and produce intermediate  $\langle key, value \rangle$  pairs; these values are written to the collector and later passed to the reducers. Reducers collect the data from collector based on group by key operation. In the group by key operation, values are grouped based on the similar key and this  $\langle key, listofvalues \rangle$  will be input to the reducer. Reducer performs aggregation or summation of these list of values. Reducer output  $\langle key, aggregatedvalue \rangle$  is written to combiner where it acts as the global reducer for all the reducers output. The combiner produces the final result which need not be in the form of  $\langle key, value \rangle$ . The driver gets the result from the combiner. Based on some condition, the driver decides about further iterations.

Even though in this work we are not using iterative MapReduce approach, we used Twister because it provides higher granularity for the *map* tasks. The configurable *map* tasks allows them to work with the large chunk of data and to produce the intermediate result. Basic Hadoop primarily focuses on fine-grained granularity for *map* tasks hence increases the size of intermediate data and causing more network latency. This is evident from the result obtained from [5], it is shown that even for non iterative (single MapReduce job) MapReduce application like CLARA, Twister outperformed Hadoop in terms of computation time. Other than this advantage, the proposed solution is generic in nature and implementable on all the frameworks which support MapReduce such as Hadoop [12], Spark [14], [1] and etc.

## 3. SCALABLE MAPREDUCE-BASED FMNN

MapReduce-based FMNN (MRFMNN) is the proposed algorithm for scaling FMNN using Twister's MapReduce framework. The input dataset is preprocessed before training the neural network.

### 3.1 Data Pre-Processing & Partitioning

The complexity of FMNN on training one object is directly proportional to the number of hyperboxes created thus far. But in the pattern space  $I^n$ , because of parameter  $\gamma$ , a hyperbox which is sufficiently away from current training object may not have any impact on the training, i.e., because of the inherent locality aspect of hyperboxes

influencing each other in the training process. One can reduce the complexity of training by considering only the near by hyperboxes instead of the entire collection without compromising the classification accuracy of the system.

With this motivation and to incorporate parallelism through MapReduce, we are proposing a space division approach in which, based on the nature of training data the pattern space  $I^n$  is subdivided. The FMNN algorithm works parallel on each subspace by working on the only subset of training data falling into that space. When we used fixed cutpoint of 0.5 in a dimension to divide the space into subspaces, the result of the division was highly unbalanced due to the scattered training objects. Hence different mappers work with the varying sizes of data. To overcome this, we used the median of the dimension as the cutpoint.

The space division approach is given in the Algorithm 1. In our approach, the user needs to provide the number of mappers that are available in the cluster. Usually, this is constrained by the number of processing cores available including all the nodes in the cluster.

---

#### Algorithm 1 DataPartition

---

##### DataPartition

**Input:** Training dataset,  $N$  : Number of Mappers  
**Output:** Partitioned Files

```

1: Randomly select  $\log_2 N$  number of dimension for dividing the space.
2: for all training dataset instances do
3:   for all  $i = 1$  to  $\log_2 N$  number of dimensions do
4:      $file_id = ""$ ;
5:     if current training object in  $i^{th}$  dimension  $<$  median of  $i^{th}$  dimension of the dataset then
6:        $file_id = Concatenate(file_id, 0)$ ;
7:     else
8:        $file_id = Concatenate(file_id, 1)$ ;
9:     end if
10:   end for
11:   Write the training object into file associated with decimal of the binary  $file_id$ 
12: end for
```

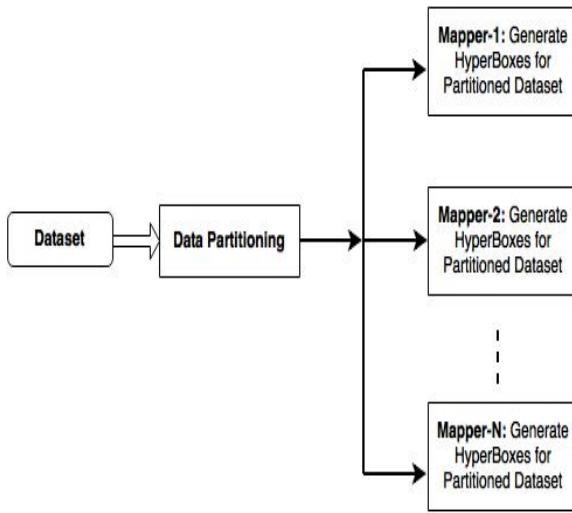
---

For the  $N$  number of mappers, an equal number of partitioned files are generated by dividing the pattern space  $I^n$  into  $N$  subspaces. The  $file_id$ 's are generated of the length of  $\log_2 N$  binary digits, hence  $N$  number of such  $file_id$ 's are generated from  $\log_2 N$  binary digits for each subspace. All the objects are written to the file of their respective  $file_id$ . During runtime, each mapper gets a single partitioned file as input. Hence, effectively all mappers work in parallel with different subspaces.

The effect of space division assures that there will not be any overlap between hyperboxes belonging to two different space partitions. This is because two points in two different space partitions satisfy to belong to on two different sides of the median cut point on at least one dimension from the set of dimensions used for space division. Hence on that particular dimension, no overlap can exist between hyperboxes constructed in these two space partitions. Hence, aggregating hyperboxes from different space partitions which are locally constructed will not affect the classification accuracy. But there is a chance for the larger number of hyperboxes being formed as there exist a possibility of the merger of

hyperboxes across the boundary of space partition in the expansion step. However, In this work, we do not consider the merging of hyperboxes across different subspaces in order to reduce the computational overhead and complexities with respect to the distributed environment. If the difference in cardinality of hyperboxes constructed through our proposed approach is similar to or slightly higher than the cardinality of hyperboxes constructed in the traditional approach, then the gains obtained in training time will compensate the disadvantage resulting from larger cardinality of hyperboxes. More importantly, this way of distribution helps in working with large datasets which can not fit into a single system's main memory.

### 3.2 MRFMNN Training



**Figure 3: MRFMNN Training- A Map Only Approach**

The flow of MRFMNN training algorithm is shown in Figure 3. The training part contains Map only approach. After generating hyperboxes locally at each mapper, the result is not aggregated at a single place since MRFMNN testing also follows distributed approach.

The partitioned training datasets will be loaded into all the mappers of Twister as the static data component, and all these mappers follow traditional FMNN training process. The hyperboxes are expanded and checked for overlap. If an overlap exists, the contraction is done to remove the overlap. All the mappers perform these processes locally and generate the hyperboxes. Once the complete network is trained, the final information of hyperboxes is written to the local disk of the respective machines.

In Algorithm 2 of MRFMNN training, we also propose a better performance logic for the FMNN. Instead of performing the overlap test on an inclusion of each training object to the network, we perform overlap and the contraction once the complete network is trained with all the training objects. In the case of the large size datasets, we observed that the total number of overlap tests resulting in contraction step is much smaller compared to the size of the training dataset. Hence, this introduces computational overhead in training process. This improved step will reduce significant computation time as it minimizes the number of overlap tests without

compromising with the accuracy of the classification.

---

#### Algorithm 2 MRFMNN Training: A Map only Approach

**Map<sub>i</sub>  $\forall i \{ i = 1 \text{ to Number of partitioned datasets}\}$**   
**Input:** Training dataset from partitioned dataset<sub>i</sub> obtained from dataset preprocessing.  
**Output:** Set of hyperboxes, minimum point  $V$ , maximum point  $W$ , and class label  $C_x$ .

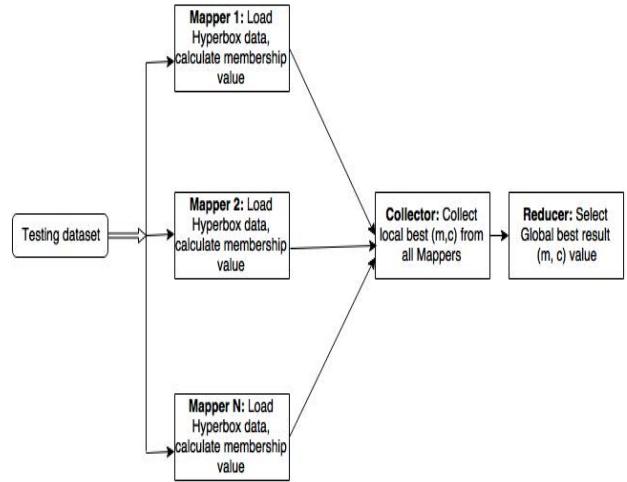
```

1: for all input dataset instances do
2:   Calculate Membership_value for all the hyperboxes
      of same decision class
3:   if Membership_Value == 0 then
4:     Create a new point hyperbox
5:   else
6:     Find a hyperbox of the same decision class which
       gives highest Membership_Value
7:     Expand the hyperbox if expansion criteria are sat-
       isfied
8:   end if
9: end for
10: For a pair of hyperboxes belongs to two different classes,
    perform overlap test. If overlap exists, perform contrac-
    tion.
11: Write all the hyperbox information to the local disk

```

---

### 3.3 MRFMNN Testing



**Figure 4: MRFMNN Testing: A MapReduce Approach**

The flow of the MRFMNN Testing is shown in Figure 4. For the given large datasets, number of hyperboxes generated are usually large in number. The complexity of classifying a test case is directly proportional to the number of hyperboxes. Hence, the MRFMNN testing algorithm also follows distributed MapReduce approach to gain speed up by utilizing the distributed resources.

An equal number of mappers are used as that of training. All the test objects are broadcasted to all the mappers. As shown in Algorithm 3, Mappers load the respective hyperbox information which was generated during the training process. The Mapper process calculates membership value for all the test objects on all the hyperboxes. The local best

result, the one which has highest membership value, and corresponding class label of that hyperbox is the output of the Map function. All the mappers for every test object write the output, i.e., local best membership value and associated class label (m,c) pair to the collector process with some dummy key.

---

**Algorithm 3** MRFMNN Testing: A MapReduce Approach  
**Map<sub>i</sub>** {  $i = 1$  to Number of partitioned hyperbox data}  
**Input:** Testing dataset  
**Output:** list of local best result, (m,c) for all test objects in dataset

```

1: Load the hyperbox information from local memory
2: for all test input data instance from testing dataset do
3:   for all hyperboxes  $1 : n$  do
4:     calculate the membership value
5:   end for
6:   Find a local best result, which has highest membership value
7: end for
8: Write the local best result (m, c) for all objects to the Collector with some dummy key

```

#### Reduce

**Input:** List of (m,c) values for each test object of test dataset from all mappers.

**Output:** Class label for test objects in dataset.

```

1: for all test objects in test dataset do
2:   Identify the global best result for all test objects, which has the highest membership value from the list
3: end for
4: Return classified class label for all the test objects to the Twister's Combiner.

```

---

Reducer gets the list of values (m,c) pairs for all test objects as the value object from the collector. The value object has the list of membership values for each test object. It filters out the value which has the highest membership value and it is considered as a global best result. Similarly, for all the test objects, the global best result is identified. Finally, Reducer writes the membership value and class label for each test object (m,c) pair as a list of values to the combiner, and the driver gets final output from the combiner.

## 4. EXPERIMENTS AND RESULTS

### 4.1 Experimental Setup

The experiments are carried out in a cluster. The experimental setup consists of 4 computing nodes. Each computing node has Intel i5 processor with clock speed of 3.2 GHz, 4 cores each, and have 4 GB of primary memory. One among the four nodes was configured as both master and worker node and remaining three were only worker nodes. These nodes were installed with OpenSuse 13.2 OS with Java 1.7.0\_75 and cluster was set up using Twister 0.9 and ActiveMQ as message broker running in one of the nodes. In all the experiments with MRFMNN, we have used 16 Mappers in the training. In testing, the same number of mappers and a single reducer is used.

### 4.2 Datasets

To check the accuracy of pattern classification of MRFMNN, standard datasets are chosen from the UCI repository

[6]. The dataset descriptions are shown in the Table 1.

**Table 1: Dataset Information**

Datasets	No of Dimensions	No of Attributes
Iris	150	4
Wine	178	13
Ionosphere	351	34
Segment	2310	19
Shuttle	58000	9
SatImage_100	600000	36
Synthetic	400000	100

To evaluate the scalability of MRFMNN on big datasets, we have extended the standard dataset Satellite Image (SatImage) from 6436 instances to 6 lakh (1 lakh = 100 thousand) instances by replicating into 100 times ( SatImage<sub>100</sub>). A Synthetic dataset is also generated which consist of 4 lakh instances and 100 attributes; this Synthetic dataset has two class labels, class 1 has randomly generated attributes ranging from [0,0.7] and class 2 has attributes ranging from [0.4,1.0]. The overlap is maintained between two classes which usually exists in real datasets also, this helps to evaluate the accuracy of classification in a more natural way.

### 4.3 Results and Analysis

For all the datasets, 70% of data is used for training the neural network and remaining 30% is used for testing the neural network. The stratified sampling is used to partition the data into training and test datasets. In both the FMNN and MRFMNN implementation, the sensitivity parameter  $\gamma$  of the membership function and the hyperbox expansion threshold parameter  $\theta$  are set to 4 and 0.3 respectively, these are experimentally tuned parameters. The MRFMNN classification accuracy is compared with traditional FMNN (sequential) for all the datasets. The results are tabulated in Table 2, the number of hyperboxes created in both traditional FMNN and MRFMNN are tabulated in Table 3.

**Table 2: Accuracy Comparison between Traditional FMNN and MRFMNN**

Datasets	Traditional FMNN	MRFMNN
Iris	95.5%	100%
Wine	96.26%	100%
Ionosphere	93.39%	91.50%
Segment	92.06%	92.92
Shuttle	99.75%	99.81%
SatImage <sub>100</sub>	90.43 %	89.56%
Synthetic	100%	100%

**Table 3: Comparison of Number of Hyperboxes Created**

Datasets	Traditional FMNN	MRFMNN
Iris	9	12
Wine	26	36
Ionosphere	59	67
Segment	23	49
Shuttle	26	50
SatImage <sub>100</sub>	226	253
Synthetic	114276	114786

Results have shown that the classification accuracy of MRFMNN is comparable to FMNN. The observation can be made that the classification accuracy of MRFMNN for the datasets Iris and Wine is improved compared to the traditional FMNN. Even though hyperboxes constructed through MRFMNN and FMNN are based on the same training dataset, the resulting hyperboxes are not identical and hence raise the differences in classification accuracies.

It is important to note that due to the splitting of the space in multiple dimensions, the number of hyperboxes created is slightly increased compared to traditional FMNN. The division of pattern space results in the split of those hyperboxes that may form across the division axes between the subspaces. However, the difference in a number of hyperboxes is not huge, indicating MRFMNN does not significantly increase complexity at the hidden layer.

Changes resulting in hyperboxes because of space division in MRFMNN is not always resulting in better classification accuracy. For example, in the case of datasets Ionosphere and SatImage<sub>100</sub> classification accuracy is slightly decreased compared to FMNN. The reasons for this will be investigated in the future.

#### 4.4 Scalability Test with Large Datasets

For smaller datasets which can be computed in a single system easily may not get computational gain with our approach. The first four datasets in Table 1 are the very small size in nature and result in a small number of hyperboxes. So the communication in MapReduce approach causes more overhead compared to actual computation. In such cases, FMNN gives better computational gain compared to MRFMNN. Hence MRFMNN is relevant for large and very large decision systems.

The scalability test with large datasets has been conducted to compare the computational gain with respect to time while training the neural network. Experiments on MRFMNN for SatImage<sub>100</sub> and Synthetic datasets are performed on single node and in the cluster of 4 node and resulting computational time for training are compared with traditional FMNN which runs sequentially on a single system. Results are summarized in Table 4.

**Table 4: Computation Time Comparison between Traditional FMNN and MRFMNN**

Datasets	Traditional FMNN (in sec's)	Space Division (in sec's)	MRFMNN 1 node (in sec's)	MRFMNN 4 nodes (in sec's)
SatImage <sub>100</sub>	9.09	13.60	5.94	2.80
Synthetic	27717	44.63	1420	1189

Based on the results in Table 4, it can be observed that MRFMNN for the SatImage<sub>100</sub> has achieved significant computational gains 34.65% over traditional FMNN in the single system alone. The computational gain has increased in the cluster environment (69.19%). Speed up of 2.12 on the single system and 3.24 on the cluster is observed. For the Synthetic dataset, speed up of 19.51 and computational gain of 94.16% is achieved between MRFMNN with the single system and traditional FMNN. But speedup for the synthetic dataset is not significantly scaled up on MRFMNN in the cluster compared to MRFMNN on a single node. Further analysis revealed that the space division in the synthetic dataset is more imbalanced across mappers compared to SatImage<sub>100</sub>.

dataset. This is resulting in the delay for reducer invocation and increased communication overhead. In future, better space division approach will be investigated to overcome this problem.

The cardinality of hyperboxes resulted by FMNN is influenced by not just the cardinality of objects and attributes but by the purity (nonoverlapping of different class regions) of the dataset and so is the influence of the dataset on the training time. Owing to this reason, because SatImage<sub>100</sub> is replicated dataset, the increase in the size of the dataset has not decreased the purity of the dataset and hence resulted in the fewer number of hyperboxes as equal to the single instance of SatImage dataset. However, the influence of replication on file I/O in space division algorithm is evident from Table 4. But in the case of dataset resulting in higher training complexity, the preprocessing time incurred for space division is negligible as evident in results obtained from the Synthetic dataset.

To analyze the computation time of testing, a comparative analysis is done for testing on MRFMNN in the cluster and traditional FMNN in a single system. A single test case of SatImage<sub>100</sub> dataset incurred 0.106 seconds in traditional FMNN whereas it took 0.626 seconds on MRFMNN in the cluster. Similarly, a test case of the Synthetic dataset with traditional FMNN incurred 8.53 seconds whereas MRFMNN took 1.59 seconds in the cluster.

The result is expected as SatImage<sub>100</sub> training results in only 253 hyperboxes whereas Synthetic dataset results in 114786 hyperboxes. Hence testing with MapReduce approach is recommended for the classification which results in a large number of hyperboxes. In the case of fewer hyperboxes, cluster communication overhead is overcompensated by actual computation.

## 5. CONCLUSION

A MapReduce-based distributed MRFMNN algorithm is proposed for the large size datasets which cannot fit in single system memory. Experiments and comparative analysis of the results have shown that our MRFMNN classification accuracy is comparable to the FMNN by achieving more computational gain. We can conclude that MRFMNN is most suitable and highly scalable classifier for massive scale huge datasets over traditional FMNN.

In future, we investigate other approaches for pattern space division to obtain the balanced subspaces such that all mappers get equal workload to obtain better speedup.

## 6. REFERENCES

- [1] Documentation| apache spark.  
<http://spark.apache.org/documentation.html>. Accessed: 2016- 07- 16.
- [2] M. A. Bhandarkar. Mapreduce programming with apache hadoop. In *IPDPS*, page 1. IEEE, 2010.
- [3] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.
- [4] J. Ekanayake, H. Li, B. Zhang, T. Gunaratne, S.-H. Bae, J. Qiu, and G. C. Fox. Twister: a runtime for iterative mapreduce. In S. Hariri and K. Keahey, editors, *HPDC*, pages 810–818. ACM, 2010.
- [5] P. Jakovits and S. N. Srivama. Evaluating mapreduce frameworks for iterative scientific computing

- applications. In *HPCS*. IEEE, 2014.
- [6] M. Lichman. UCI machine learning repository.  
<http://archive.ics.uci.edu/ml>, 2013.
- [7] D. Ma, J. Liu, and Z. Wang. The pattern classification based on fuzzy min-max neural network with new algorithm. In J. Wang, G. G. Yen, and M. M. Polycarpou, editors, *ISNN (2)*, volume 7368 of *Lecture Notes in Computer Science*, pages 1–9. Springer, 2012.
- [8] M. F. Mohammed and C. P. Lim. An enhanced fuzzy min-max neural network for pattern classification. *IEEE Trans. Neural Netw. Learning Syst.*, 26(3):417–429, 2015.
- [9] A. V. Nandedkar and P. K. Biswas. A fuzzy min-max neural network classifier with compensatory neuron architecture. *IEEE Trans. Neural Networks*, 18(1):42–54, 2007.
- [10] P. K. Simpson. Fuzzy min-max neural networks. i. classification. *IEEE Trans. Neural Networks*, 3(5):776–786, 1992.
- [11] J. Vieira, F. M. Dias, and A. Mota. Neuro-fuzzy systems: a survey. In *5th WSEAS NNA International Conference*, 2004.
- [12] T. White. *Hadoop: The Definitive Guide*. O'Reilly Media, 2. auflage. edition, 11 2010.
- [13] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.
- [14] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster computing with working sets. In E. M. Nahum and D. Xu, editors, *HotCloud*. USENIX Association, 2010.
- [15] H. Zhang, J. Liu, D. Ma, and Z. Wang. Data-core-based fuzzy min-max neural network for pattern classification. *IEEE Trans. Neural Networks*, 22(12):2339–2352, 2011.

# Feasibility of Fog Computing Deployment based on Docker Containerization over RaspberryPi

Paolo Bellavista

Dept. Computer Science and Engineering (DISI)  
Alma Mater Studiorum - University of Bologna  
40136 - Bologna, Italy  
[paoletto.bellavista@unibo.it](mailto:paoletto.bellavista@unibo.it)

Alessandro Zanni

Dept. Computer Science and Engineering (DISI)  
Alma Mater Studiorum - University of Bologna  
40136 - Bologna, Italy  
[alessandro.zanni3@unibo.it](mailto:alessandro.zanni3@unibo.it)

## ABSTRACT

Fog computing is strongly emerging as a relevant and interest-attracting paradigm+technology for both the academic and industrial communities. However, architecture and methodological approaches are still prevalent in the literature, while few research activities have specifically targeted so far the issues of practical feasibility, cost-effectiveness, and efficiency of fog solutions over easily-deployable environments. In this perspective, this paper originally presents i) our fog-oriented framework for Internet-of-Things applications based on innovative scalability extensions of the open-source Kura gateway and ii) its Docker-based containerization over challenging and resource-limited fog nodes, i.e., RaspberryPi devices. Our practical experience and experimental work show the feasibility of using even extremely constrained nodes as fog gateways; the reported results demonstrate that good scalability and limited overhead can be coupled, via proper configuration tuning and implementation optimizations, with the significant advantages of containerization in terms of flexibility and easy deployment, also when working on top of existing, off-the-shelf, and limited-cost gateway nodes.

## Keywords

Fog Computing, Internet of Things, Gateways, RaspberryPi, Kura Framework, Resource Management, Container, Docker.

## 1. INTRODUCTION

The Internet of Things (IoT) is pushing the widespread diffusion of an unprecedentedly huge number of connected devices, by introducing new classes of applications, expected to have potentially massive impact via the involvement of communicating sensors and actuators. At the same time, cloud computing techniques are nowadays well developed and represent an

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [Permissions@acm.org](mailto:Permissions@acm.org).

ICDCN '17, January 04-07, 2017, Hyderabad, India © 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007777>

industry-mature technology that well complements IoT, with all the potential to assist and support the deployment and execution of IoT applications. In fact, the cloud provides IoT applications with new opportunities to be industrially feasible and cost-effective, e.g., via inexpensive storage of time series of monitoring data and via resource elasticity in response to critical situations. However, the related literature is recognizing that IoT cloud-centric architectures consisting of only two layers (i.e., sensors/actuators and the cloud) are inadequate in many IoT application domains and deployment scenarios [1, 2, 3], for example because they tend to assume high sensor-to-cloud bandwidth and stable connectivity. In addition, IoT devices-to-cloud latency is not always compatible with application requirements and connectivity costs are not negligible in several business cases.

In the above situations, it starts to be recognized that at least an additional layer, e.g., composed by gateway nodes relatively local to sensors/actuators, can significantly enrich the flexibility and suitability of the IoT-cloud architecture. This intermediate layer may provide and support in: scalability via more distributed and localized processing/state maintenance; data aggregation, with gateways working as data sinks for devices working with high sample rate; interoperability at network edges by overcoming the possible heterogeneity due to the large variety in integrated sensors and actuators; complementing resource-constrained IoT devices, usually with few storage/computing/communication resources and limited energy power; dynamic and efficient registration/discovery of IoT devices. It is manifest that the effective and efficient integration of IoT and the cloud is technically challenging and represents a very hot research topic nowadays. Starting from seminal approaches of proxy intermediation/optimization even before the cloud introduction [4, 5, 6], several more recent related papers propose, with slightly different flavors, three-layer IoT-cloud architectures with the above intermediate layer of resources at network edges, by naming their approach as fog computing [7, 8], cloudlet [9], edge computing [10], or follow-me cloud [11]. Below we will use the term fog computing in a general way to indicate any three-layer IoT-cloud architecture with an intermediate layer of geographically distributed gateway nodes, typically well positioned at the edges of network localities that are densely populated by IoT sensors and actuators.

Fog computing is demonstrating to be a powerful paradigm to overcome several technical challenges of IoT applications; however, some design/implementation/deployment issues, both application-specific and general-purpose across different

application domains, still have to be addressed to turn effective fog solutions into reality [12]. Moreover, the fog-cloud integration literature is still in its infancy.

Therefore, in order to promote a further step of advancement of the research in the field, this paper presents how we have designed, implemented, and experimented an innovative fog computing solution based on two primary directions of gateway node improvement, i.e., i) scalability extensions of the IoT gateway provided by the open-source Kura framework [13] (Kura is currently considered one of the most impactful and industry-mature open-source project for IoT-cloud integration and management), and ii) Docker-based containerization [14] over resource-limited RaspberryPi devices. On the one hand, as better detailed in the following parts of the paper, we have originally extended the Kura framework with the introduction of local brokers for scalability purposes. In fact, the regular Kura IoT gateways simply aggregate all the data gathered by their sensors and send them to a Message Queue Telemetry Transport (MQTT) broker running on the cloud, by delegating all the associated computations to geographically distributed resources on the global cloud. On the contrary, our prototype relevantly extends the Kura IoT gateway to make it the central component of a fog infrastructure where fog nodes can scalably serve as local MQTT brokers and dynamically coordinate among themselves, also without the need of the continuous support of global cloud resources. On the other hand, we significantly enrich the fog intermediate layer by giving the opportunity of exploiting container-based virtualization on top of IoT gateways, with full infrastructure support (download, update, and management of virtualized images based on industry-mature Docker). To the best of our knowledge, this is one of the first cases of implementation and experimentation of virtualization techniques over fog nodes, in particular while working with IoT gateways with very limited resources, such as RaspberryPi nodes. Container-based solutions for the fog offer the well-known advantages of abstracting implementation details, more easily achieving interoperability and portability between possibly heterogeneous fog nodes, etc., similarly to the adoption of virtual machines, but with lightweight migrations and better performance [15, 16]. We claim that fog solution containerization can significantly leverage the introduction and diffusion of fog computing techniques in mature deployment scenarios, by allowing more automated and easier integration and installation, in particular over large-scale and industrial execution environments.

In addition, the paper is original in reporting qualitative and quantitative results about our deployment and experimentation experience over a real in-the-field execution environment, where fog nodes are run over low-cost and off-the-shelf RaspberryPi devices, which exploit Docker-based containerization to dynamically install the needed fog module functionality only when and where needed. In particular, our work demonstrates the feasibility of using even extremely constrained nodes as fog gateways: good scalability and limited overhead can be achieved, via proper configuration tuning and implementation optimizations, also when using Docker over RaspberryPi IoT gateways; the reported performance results show that the Docker-based containerization is usable in several IoT application domains (where we can accept some latency in the cases of need for re-deploying new/extended/modified fog features on selected IoT gateways), with significant advantages in terms of flexibility and easy deployment, thus leveraging more rapid diffusion of fog

solutions also in existing deployment environments.

The remainder of the paper is organized as follows. Section II highlights our primary original solution guidelines for IoT gateways working as full fog nodes, by extending the exiting base of the open-source Kura IoT gateway. Section III describes how we have originally employed containerization techniques, in particular Docker, to dynamically configure fog nodes in a highly portable and standardized way. Implementation insights, quantitative performance results, and lessons learned from our practical experience of deployment and experimentation are reported in Section IV, while conclusive remarks and directions of ongoing work close the paper.

## 2. EXTENDING THE KURA IOT GATEWAY FOR SCALABLE FOG

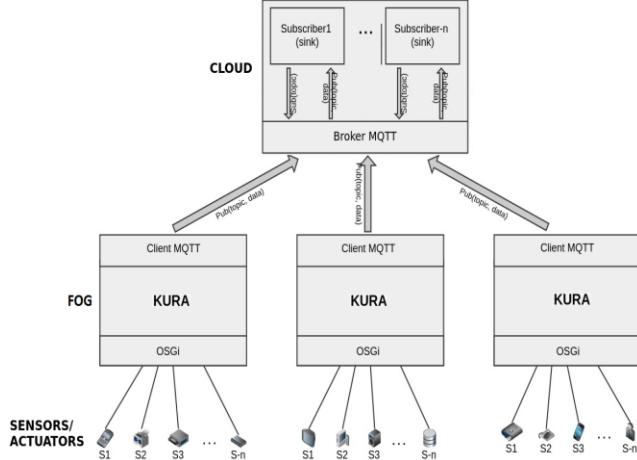
In this section we highlight our original fog computing middleware solution based on the open-source Kura framework. In particular, we describe how the Kura architecture exhibits some non-negligible weaknesses for a scalable utilization in fog computing scenarios, mainly due to its direct IoT devices-to-cloud model of interaction. This has motivated our extension work towards IoT gateways capable of acting as local MQTT brokers and of coordinating among themselves according to both cluster and mesh organizations, as detailed in Sections 2.2 and 2.3, respectively.

### 2.1 The Kura Framework

Kura is an open-source framework for IoT applications that provides a platform for building IoT gateways. In particular, it abstracts the design and implementation of gateways from the complexity of real-world industrial scenarios consisting of heterogeneous hardware/network devices. To this purpose, Kura aggregates and controls device information, as well as it supports the simplification of the overall development and deployment process. Kura is based on Java OSGi to support, in a widely accepted way, dynamic management of software components with no need of operation suspension, to simplify the process of writing reusable software building blocks and to create self-contained pluggable packages (i.e., bundles) specifically suitable for IoT applications. In particular, we have decided to employ Kura as our implementation basis because it can serve as a suitable container for machine-to-machine applications running in service gateways. Kura uses MQTT [17] as its central protocol: it provides different features for message publication towards MQTT brokers and the subscription to specific broker-supported topics. In addition, Kura components can support routing functionality, by managing wired/wireless communications and by allowing VPN connections, firewall usage, and NAT operations. Kura APIs offer easy access to the underlying hardware including serial ports, GPS, watchdogs, USBs, gpio-s, etc. Finally, Kura is open-source, with a fervent and growing community supporting the initiative and continuously proposing extensions and innovative Kura-based solutions, thus guaranteeing maintenance/evolution support for the years to come (very relevant for supports and applications of industrial interest). Figure 1 shows the base IoT-cloud architecture pushed by the existing Kura framework, with the “regular” way of using Kura in such an architecture.

By delving into finer technical details, by referring to the default Kura architecture, in our research work we have found and shown some non-negligible weaknesses that motivate the need for some

relevant extensions, at least in the perspective of efficient fog computing exploitation:



**Figure 1. A Cloud-integrated Architecture based on Kura**

- *Single MQTT broker on the cloud.* Kura allows to communicate only with a single broker located on the cloud and, thus, tends to send all the collected data from IoT devices to that broker. This raises non-negligible concerns, such as i) the single broker deployment option may cause a performance slowdown in case of high load, enabling only to deploy small applications with a limited number of devices or with very limited sensor sampling rates; ii) Kura gateways can only dispatch sensed information towards the cloud, with no fog-oriented processing operation performed locally; and iii) Kura exploits persistent sockets to connect its gateways to the integrated cloud, with a non-negligible waste of resources in case of intermittent/interrupted communications or of sporadic interaction;
- *Flat topology.* Gateways are usually hosted on hardware equipment with a limited amount of resources, hence a single gateway or a set of gateways organized in a flat topology can perform only relatively limited operations, i.e., processing, storage, inference, etc., which might be insufficient for several application domains of interest and in many envisioned IoT applications.

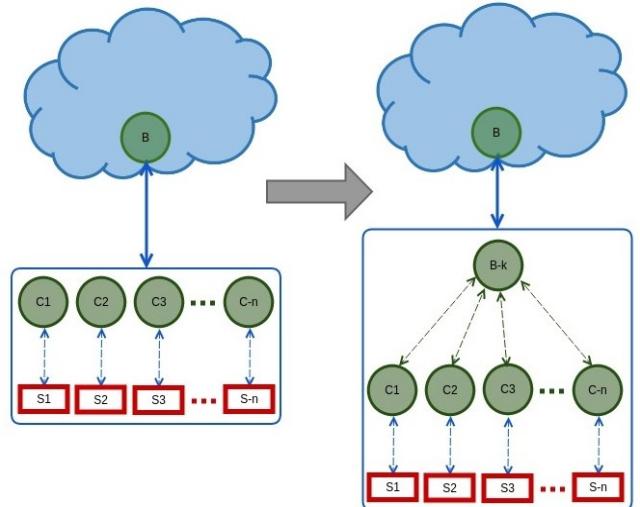
The following sub-sections try to concisely illustrate how we have extended Kura to overcome the above weaknesses, thus making it more suitable as a basic building block for fog computing infrastructures to support three-layer cloud-integrated architectures for IoT and consequently wide-scale IoT applications.

## 2.2 Gateway-side MQTT Brokers

Our first relevant extension is towards the inclusion of an MQTT broker, e.g., Mosquitto [18], on each gateway, as depicted in Figure 2, in order to collect sensed information at the gateway side and, at a later stage, possibly after local processing and inferencing, to send filtered/processed/aggregated data to the cloud. The main advantages deriving from this extension, in the fog computing perspective, are:

- *Enabling hierarchical topologies.* The internal topology

turns, from a flat structure where all the clients can send data to the cloud-side broker, to a hierarchical structure where the gateway-side broker can serve as a local root. Further extensions of the hierarchy can consider to use multiple brokers or more client/broker levels to further strengthen the before-the-cloud processing capabilities. This potential augmentation of the infrastructure should be dynamically fitted, however, against the currently available resources at the IoT gateway side, thus further pushing for the opportunity of advanced dynamic deployment support;



**Figure 2. Adding Gateway-side MQTT brokers**

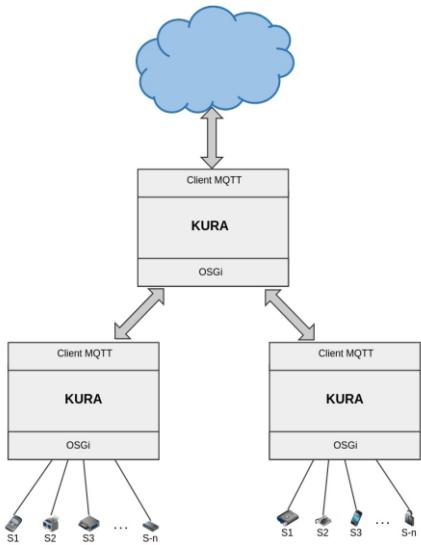
- *Gateway-level MQTT message aggregation.* It is possible to perform and implement typical fog computing functionality on the gateway, e.g., basic data aggregation and filtering actions, Small Data processing [19], and alike. Our extension still exploits the standard MQTT protocol, as the original Kura framework, but in two following separated segments, i.e., sensors-to-gateway and gateway-to-cloud-hosted brokers;
- *Real-time message delivery and reactions.* The introduced ability of our extended IoT gateways to collect data and perform actions can significantly speed up system reactions. Reactions with hard/soft real-time constraints may then be delegated to IoT gateway components closer to the edges of the network, with improvements in time and quality of reactions (fresher sensed data);
- *Actuation capacity and message priorities.* A peculiar ability of fog computing is to dynamically determine the situations when it is necessary an immediate actuation or when it is possible to send data to the cloud for intense postponed analytics. We propose that each IoT application, in relation to the context and the type of information sensed from the environment where it is immersed, must define sets of messages with the relative priority and consequently differentiating the type and the promptness of reactions, as well as its desired level of fog-cloud interplay. For instance, in smart city-oriented vehicular applications, cloud analytics are particularly useful and tight coupled outside vehicle endpoints for the evaluation of traffic patterns and to detect

best ways to reach destinations. However, this is limitedly related to operations inside vehicles (intra-vehicle applications) where fog nodes have to deal more frequently with real-time actions;

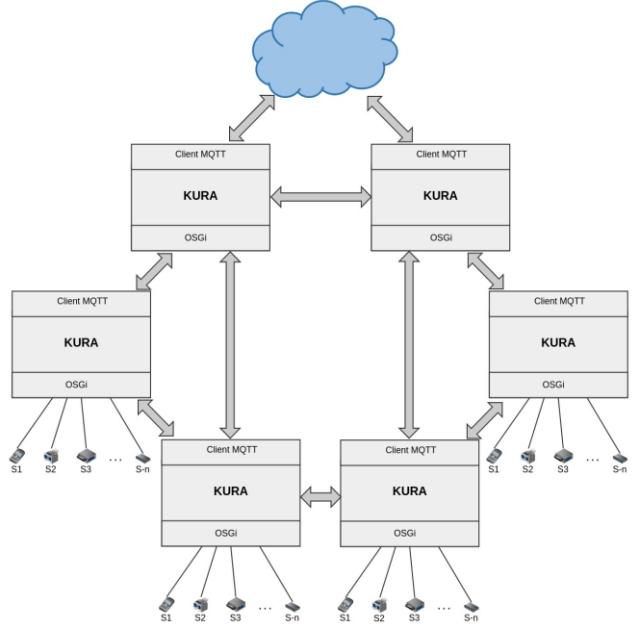
- *Locality awareness and locality-oriented optimizations.* The information sensed by IoT devices are processed by each gateway in its locality and thus with a usually more complete knowledge of the local environment from which the sensed data are generated;
- *Gateway-cloud connection optimization.* The cloud-side broker is no more required to stay alive at any time and to continuously receive sensed data collected from each part of the overall deployment environment. Given that it receives data after that gateways have performed processing/aggregation/filtering/... operations over them, the usage of persistent sockets is often not required and inefficient (persistent sockets are the most widespread communication mechanism in available MQTT brokers). Therefore, in our extension, we have decided to exploit non-persistent sockets, which are dynamically established only when necessary, to exchange data between the cloud-hosted broker and the lower-level brokers at IoT gateways.

### 2.3 Enabling Cluster/Mesh Topologies for Kura Gateways

Another significant IoT gateway extension that we originally propose relates to virtually strengthening, in a dynamic way, the available gateway resources via the combination of multiple physical gateways and the aggregation of their resources. For example, Figure 3 depicts a possible cluster-oriented physical topology for Kura gateways realizing a virtual and strengthened higher-layer gateway, while Figure 4 shows a similar concept through virtualization of an underlying mesh topology. In fact, the support to a more powerful intermediate layer, based on either cluster or mesh topologies, is demonstrating to be central to enable scalability in large IoT applications.



**Figure 3. The Supported Cluster Organization of IoT Gateways**



**Figure 4. The Supported Mesh Organization of IoT Gateways**

We claim that the most significant advantages associated with full and seamless support to cluster/mesh organizations of IoT gateways are:

- *Kura gateway specialization.* The cluster/mesh topology consists of different gateways with different properties and each gateway is more suitable to perform some tasks or to cover some functionalities rather than others. For example, some gateways may be deployed to be more specialized to aggregate data, others for context-aware processing/filtering operations, others for locality-efficient data storage and indexing, and so on;
- *Locality exploitation and data quality.* Locality-based optimizations can be improved by performing more accurate and complex analytics and by taking advantage of larger resource availability;
- *Geo-distribution* is another feature whose performance/effectiveness advantages depend on the number of interworking gateways (and their total amount of available resources). With the increased number of cluster/mesh-organized gateways it is possible to manage dense sensor localities and to make the overall distributed deployment scale better;
- *Scalability.* Cluster/mesh of gateways can scale more easily, according with the total amount of resources available, also by facilitating the realization of load balancing operations and dynamic failover management;
- *Security and privacy.* Security and privacy improvements are connected to locality-based operations and full locality visibility, which allow having more complete and accurate knowledge of the environment where gateways are operating, towards the creation of more efficient support

features. At the same time, the proximity between environment and data elaboration places, partially moved to the layers of gateways, can improve privacy and simplify decentralized ownership management.

### 3. LEVERAGING FOG DEPLOYMENT VIA CONTAINERIZATION

We claim the suitability and effectiveness of introducing a highly manageable and interoperable way to create fog nodes on-the-fly via the adoption of containerization techniques, whose advantages are deemed prevalent to disadvantages also in the case of IoT gateways with limited resource availability. In particular, this section presents how we have enhanced our fog computing middleware via dynamic IoT gateway configuration through i) the creation of standard gateway base configuration; ii) the creation of container-based (typically small and atomic) applications/services, each with very specific functions; and iii) the dynamic orchestration of fog middleware services by the global cloud, with the possibility to install, replace, or extend the currently installed configurations and available middleware services.

#### 3.1 Overview and Motivations

Containers are nowadays a very effective alternative solution to more traditional Virtual Machines (VMs), also allowing the deployment of virtualized resources on less powerful server hosts than in regular cloud computing [15], with relatively limited performance impact. Container-based virtualization is an industrially mature technology that provides real virtualization at the OS level, rather than a full OS on virtual hardware, with all the primary virtualized properties achievable via VMs, e.g., own network interfaces, own filesystem, isolation in terms of security and resource usage, but with much more lightweight operations. Since container-based services and applications share their underlying OS, the associated deployments are significantly smaller in size than VM-oriented hypervisor deployments, thus making it possible to store hundreds of containers on a physical host, as well as restarting a container without rebooting the OS, which is very relevant in several application domains [20]. The work described in [21] highlights how containers, differently from VMs, are more flexible for packaging, delivering, and orchestrating both software infrastructure services and application-level components, i.e. typically for tasks performed by a PaaS. In this paper, we take this general idea and perspective, and originally extend it to the applicability domain of fog computing middleware. In addition, containers provide an abstraction that makes each container a self-contained unit of computation [22], thus allowing easier portability and interoperability, with lightweight components and good suitability for distributed applications. Several tools provide the abstraction and implement the container-oriented models, e.g., LinuX Containers (LXC), Docker, CRIU, systemd-nspawn, rkt, runC, OpenVZ, and many others.

##### 3.1.1 LXC

LXC (LinuX Container) [22] is a virtualization technology to create multiple Linux virtual environments, which provides low-level kernel features to guarantee sandboxing processes and to control resource allocation. LXC provides the isolation of processes on the shared OS via kernel namespaces, which are the basic isolation mechanism to separate containers in LXC, and via control groups, which allocate and manage container resources.

##### 3.1.2 Docker

Docker is an industry-popular and mature initiative to build independent containerized applications by extending LXC with a high-level tool, along with an easy to use interface and deployment process. For instance, among the others, Docker tries to overcome dependency issues by packaging each component and its dependencies in order to solve conflicting or missing dependencies and to overcome platform differences by the Docker engine. The available Docker implementation provides:

- *portability*. It defines a format for bundling an application with all its dependencies into a single object, benefits from LXC process sandboxing, and extends it with an abstraction for machine-specific settings, e.g., networking, storage, logging, distribution, etc.;
- *component reuse*. Any container can be used as a base image to create more specialized components;
- *versioning*. It supports the tracking of container versions, by inspecting the differences between versions automatically, as well as committing new versions and rolling back. This enables the possibility to perform only incremental uploads/downloads based on differences between two versions, with limited bandwidth usage;
- *a large set of supporting tools*. Docker defines APIs for automating and customizing the creation and deployment of containers and many tools may be integrated to extend its capabilities;
- *container sharing*. There is the rich support to a public registry where anyway can upload/download containers;
- *application orientation*. Docker is optimized for the deployment of applications, rather than LXC that is more OS-oriented with focus on containers as lightweight machines;
- *automatic build mechanisms*. Docker includes a tool to automatically assemble a container from source code, with full control over application dependencies, build tools, packaging, and so on.

In short and roughly speaking, Docker is gaining momentum, in both the academic and the industrial communities, because it provides a high-level API and good overall performance, by adding a limited overhead if compared with LXC containers.

##### 3.1.3 CRIU

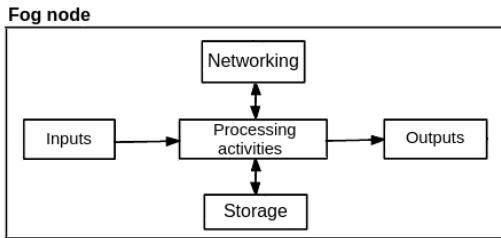
CRIU [23] is an emerging software tool, used to checkpoint/restore a tree of running processes: CRIU can easily freeze a running application, or part of it, and checkpoint it to persistent storage as a collection of files. CRIU can automatically enrich the code of running containers in order to get the dump of the state container when proper triggers are fired; associated with that, a typical and interesting CRIU employment nowadays is as the base for implementing container live migration.

We have decided to adopt Docker as the containerization technology for our original solution described below because it provides a high-level extension of LXC capabilities, by guaranteeing at the same time a reasonably lightweight usage of resources, with performance results that are comparable with LXC. In addition, Docker is based on a fervent large community of developers, thus being a significant advantage if compared to competitors. Moreover, CRIU has not been considered because at the moment it only supports process migration (not a full-support containerization solution).

## 3.2 Configuration and Management of IoT Gateways

### 3.2.1 Gateway standard base configuration

We define a standard gateway configuration that is the base for our fog-oriented IoT gateway. In our proposal each fog participant is compliant with that configuration in order to promote the deployment of general-purpose fog nodes, which can then be dynamically extended with additional and optional middleware services. To this purpose, we have defined macro-operations that compose the skeleton of any application powered by our fog computing middleware. Namely, in our proposed solution it is possible to define the following base macro-operations: I/O input operations, storage facilities, service computation, networking capabilities, and output operations. The base configuration must only manage the macro-operations flow (we call this skeleton in the following) and system lifecycle from a high-level and abstract point of view in order to push for and facilitate openness and portability. For example, every fog node must specify the macro-services list and the skeleton to follow to achieve the final desired behavior, as shown in the simple example in Figure 5.



**Figure 5. Fog node skeleton**

In this way every fog node has the same base configuration and the same skeleton, with no relationships on how the macro-service will be composed further in single services and their specific implementations. In addition, the base skeleton does not give any indication about the applications that will use the supporting fog node as their IoT gateway (dynamic association).

### 3.2.2 Container-based services

In our IoT gateway design and implementation, we have defined macro-service composition and the single service implementation through containers. Contrarily to macro-services, single services are specific to the application domain where they are used in. We propose a process in which a system administrator can build a complete functionality, e.g., data aggregation, data filtering, data normalization, data processing, database access, output management, etc., within a container, can package it for deployment, and then have it sent to the needed IoT gateways. In this way, by exploiting application compositability, it is possible to configure a given IoT gateway in relation to the specific application(s) where it is used, by loading on it all and only the containers related to such specific application(s).

In addition, every time a new component is implemented or a new version of a functionality is released, we can upload to the gateway the new container-based version of the needed functionality. Moreover, each container may contain a time-to-live value, particularly suitable for long-term container management, that specifies the validity of a container version: if not refreshed, the expired version of the container is automatically discarded in our fog middleware solution.

Each containerized component is isolated and independent from the others, thus, it is possible to update and/or upload only specific functions/containers in a fully independent way. In addition, we can use every node interchangeably to create an application because all the nodes have the same skeleton (skeleton-based homogeneity). This property allows to dynamically create fog nodes for a particular application, e.g., in relation to hardware-specific properties of an IoT gateway and to application relevance/priority in a given time interval. For instance, for compute-intensive applications we can employ more compute-powerful nodes, rather than storage-intensive applications where we need to enhance database operations, or I/O intensive applications where the focus is on communication capabilities. In addition, some applications are particularly used in relation to the time of the day, with rush hours with high peak load conditions and the need of more gateways (or resources on the available gateways), and off-peak times when many resources may be released. For example, smart traffic light systems have quite predictable stress patterns, by permitting to optimize IoT gateway resource consumption and deployment by sharing them with other applications far from rush traffic hours.

### 3.2.3 Management and orchestration

In our fog computing middleware, container management and orchestration are performed by the cloud computing layer, by taking advantage from its potentially global visibility and from its key ability to perform predictive long term data analysis, thus trying to capture system behaviors and to infer system evolution [11]. In fact, in this perspective, cloud computing has the global view of resource distribution and application usage that allows to better manage the efficient management and orchestration of the distributed containers over the fog nodes.

We claim that a cloud orchestrator is necessary to deal with multi-containers management to handle: i) the creation of the cluster of containers that compose an application; and ii) the distributed deployment of an application over several IoT gateways. Container scheduling loads containers into multiple gateways, indicating how to run them, e.g., in which order to run containers, their dependencies, the minimal resources needed, or grouping all containers for a specific application for optimized transmission to gateways. More advanced tools may check if the destination hosts are correctly configured to run containers, by possibly trying to solve the associated issues. The combination of containers (which abstract from platform heterogeneity) and orchestration mechanisms (which enable logically centralized container management) allows to create a loosely coupled architecture between the configuration capability and the underlying middleware. This promotes the reuse, automation, and decoupling of allocation algorithms from deployment, by allowing to focus more on which tasks to send to the IoT gateways instead of how to deploy it on the target platform. In order to integrate container management and orchestration functions in our fog computing middleware, we have thoroughly evaluated several existing solutions in the literature, e.g. Docker Swarm, Kubernetes, Apache Mesos; the rich set of existing container-oriented orchestrators is a manifest sign of the academic/industrial interest in the approach we have decided to apply to fog computing nodes.

### 3.2.4 Docker Swarm

Docker Swarm [24] is a native tool to manage Docker clustering. It is easy to use and very flexible; in addition, it exposes Docker API to be used by other Docker tools, e.g., Docker Compose.

Docker Swarm is a relatively novel tool, that is developing quickly and gaining momentum, but also still needs further improvements to support complex scheduling, to be used in a production environment or for a large-scale system due to some limits. For instance, at the moment, Docker Swarm does not support any sophisticated load-balancing mechanism and lacks interoperability with other industrial tools in the field.

### 3.2.5 Kubernetes

Kubernetes [25] is an orchestration framework, typically used as a clustering engine to define containers organization within an application. Kubernetes has demonstrated to achieve good performance and in a scalable way, without adding significant overhead to existing containers. In addition, thanks to its plugin architecture, it easily integrates with different vendors' tools and technologies. In addition, Kubernetes self-healing, auto restarting, replication, and rescheduling mechanisms make it more robust and suitable for container-based applications [15]. Primary Kubernetes drawbacks relate to complexity because the setup is quite complicated and installations differ from platform to platform.

### 3.2.6 Apache Mesos

Apache Mesos [26] is a low-level, portable, and very reliable orchestrator. It has been designed to achieve high performance and to scale to very large clusters, even if it has high resource overhead. While being very interesting for other deployment scenarios, due to its overhead load it is not recommendable for usage in some IoT application domains and in fog computing middleware.

Let us note that, even if not detailed here and out of the specific scope of this paper, in our fog computing middleware we have decided to integrate Docker Swarm as the provider of the basic container orchestrator mechanisms. In fact, we claim that, at the moment, Docker Swarm represents the best tradeoff between good industry-level solidity, high performance, and limited overhead: in our preliminary experimental work it has demonstrated to provide good overall performance; it is improving rapidly, with a relatively large community of users behind it; from the perspective of resource consumption, it has shown to be more lightweight than Mesos and Kubernetes. We recognize that Mesos and Kubernetes can provide more complete and robust orchestrators, not only limited to Docker containers; however, at the moment, we claim the suitability of a more lightweight solution, also easier to be integrated with additional mechanisms and policies, specifically designed for the targeted goal of orchestrating distributed fog resources based on global visibility of application deployment environment.

## 4. PRIMARY IMPLEMENTATION INSIGHTS AND PERFORMANCE RESULTS

Here, for the sake of brevity, we describe some primary implementation insights of how we use containerization within our fog computing middleware and we report some related experimental results, both in standalone and multi-containers deployment scenarios, when using fog nodes with very limited resource availability as IoT gateways; the goal is to quantitatively evaluate the scalability of the approach and its introduced overhead in a particularly challenging deployment environment. Given that, in our experimental work, we have noticed a relevant dependence of performance indicators on the different storage

drivers exploited, especially in the kind of IoT applications used in our testbed (see Section 4.2), here we start by considering which different storage drivers are supported to be plugged in with Docker containers (Section 4.1). Additional implementation insights about our broker-related IoT gateway extension are omitted here for the sake of brevity; the interested reader is directed to [http://lia.disi.unibo.it/Research/\\_MQTT\\_CoAP\\_Integration](http://lia.disi.unibo.it/Research/_MQTT_CoAP_Integration) for additional implementation details and associated performance results.

## 4.1 Filesystem Selection and Impact

Docker has a pluggable storage driver architecture to give the flexibility to integrate and configure the most suitable storage driver for the specific application requirements and deployment characteristics, ranging in a relatively wide set of different technologies [14], the most widespread of which are AUFS, Device-mapper, and OverlayFS.

AUFS [27] is a layered filesystem that can transparently overlay one or more existing filesystems, by merging multiple layers into a single representation of a filesystem [28]. This allows to reuse layers, i.e., multiple containers that require the same base image, and to support efficient versioning of the used images, i.e., by including and exporting only differences between different versions of the same image. AUFS uses the Copy-on-Write (CoW) technology that creates a snapshot of a file every time a process needs to modify it. AUFS, based on CoW support, has demonstrated to have a non-negligible overhead in case of large size files or with high numbers of folders; however, its performance can be tuned and optimized dynamically via different configuration mechanisms, which typically are used to achieve maximum performance for container creation and I/O operations [28].

Device-mapper introduces (and works at) a further layer, i.e., block level: it is based on the so-called thin provisioning [29] where blocks realized via a sparse file allow to use Docker in a very easy-to-use way with no need for static configuration definition. The primary drawback is related to associated performance: every time a container writes to a block in the allocated pool of blocks, this block should be copied to the sparse file, thus introducing non-negligible latency.

OverlayFS [30] is a filesystem based on two main layers: the upper filesystem layer is visible to applications and readable/writable; the lower filesystem modules are instead not visible and read-only; they realize the base layer to be merged with the upper layer (containing latest data modifications) to have the final updated vision. OverlayFS provides good performance thanks to page caching and the two-layers design: for example, depending on its architecture, AUFS tends to introduce more latency due to the need of searching among more layers. However, OverlayFS can exhibit significant overhead in the case of high dynamicity and high numbers of modification operations.

Finally, for the sake of completeness, let us rapidly sketch a very short description of the less popular BTRFS and VFS. On the one hand, BTRFS, similarly to the AUFS driver, is based on CoW and offers better scalability and reliability in cases of non-extreme modifications rate. On the other hand, in VFS each layer is implemented as a different folder because VFS does not support CoW, thus leading to a simple and very portable solution, but with non-negligible drawbacks in terms of achievable performance.

## 4.2 Containerization Overhead Performance Results

As the realistic testbed for our fog computing middleware solution, we have developed and deployed a simple use case application in the domain of Smart Connected Vehicles (SCV). In our SCV application, a fog node located on a smart bus (following pre-determined routes) assists the other distributed SCV participants; for the sake of simplicity, in the example illustrated here to show the performance results of our fog computing containerization, the fog node has the main goal of elaborating and aggregating the data gathered from multiple sources. In other words, the fog node acts as a mobile sink collecting data from a dynamically determined set of heterogeneous sensors, that can be installed on-board of the transit vehicles or on other infrastructural components, e.g., base stations with car detection modules at road crossings; the fog node keeps track of gathered data by storing and sorting information, e.g., according to priorities, originating locations, previous warnings or error signals, and so on. In addition, the mobile sink can decide to spread valuable concise information to other SVC participants opportunistically encountered during its travel, thus contributing to overall system awareness and to the emergence of cooperative behaviors.

To this purpose, we have implemented and experimented an application running on our fog node that behaves according to the skeleton of i) gathering data from sensors, ii) storing gathered information, iii) periodically (for energy consumption motivations) reading local storage, and iv) processing the read data to identify alerts/warnings. The application is implemented in Golang, which is the most supported language in Docker, and is an example of application that, in different time intervals, has relevant requirements in terms of both computing and I/O operations. As already anticipated, to collect quantitative performance indicators in a particularly challenging deployment environment, we have used for our fog nodes some RaspberryPi 1 Model B+ devices, with 512MB RAM and a single core CPU.

Here we report some interesting results about our experimental work by focusing on Docker-based containerization and on the

impact of filesystem configuration on performance. In particular, we compare the performance of single containers in relation to native code execution, by estimating the overall overhead and delays introduced by the exploitation of Docker-based containers with either AUFS, Device mapper, or OverlayFS. In addition, we have thoroughly evaluated the usage of multiple concurrent containers over a single fog node (which is highly typical in envisioned future applications), also to verify the capability of even very constrained devices to manage several containers simultaneously. All the results reported in the following do not include the latency and effects of orchestration mechanisms because, in this paper, we wanted to exclusively focus on the performance characterization of the extended IoT gateway on the fog node.

Table 1 reports the most significant results in terms of average time, over a wide set of 100 runs, to complete the execution of our sample SVC application in the case of native code execution vs. container-based execution with the exploitation of different storage drivers, i.e. AUFS, Device mapper, and OverlayFS. In addition, the table highlights the time contributions associated with the most relevant operations performed during the SVC application execution: container creation time; I/O operations (file opening and reading – size of 1.6 MB); CPU operations (double sorting, first alphabetically and then by words length). The maximum standard deviation experienced for the reported results has been of 0.2 s.

Operation category	native	Docker + AUFS	Docker + Device mapper	Docker + OverlayFS
Start Container	-	3.5 s	9.1 s	3.3 s
I/O Operations	1.6 s	4.3 s	4.7 s	4.3 s
CPU Operations	3.1 s	3.4 s	4.2 s	3.5 s
Total Execution	4.7 s	12.5 s	21 s	11.8 s

Table 1. Native-code and container execution time

The build time of the container-based version of the SVC

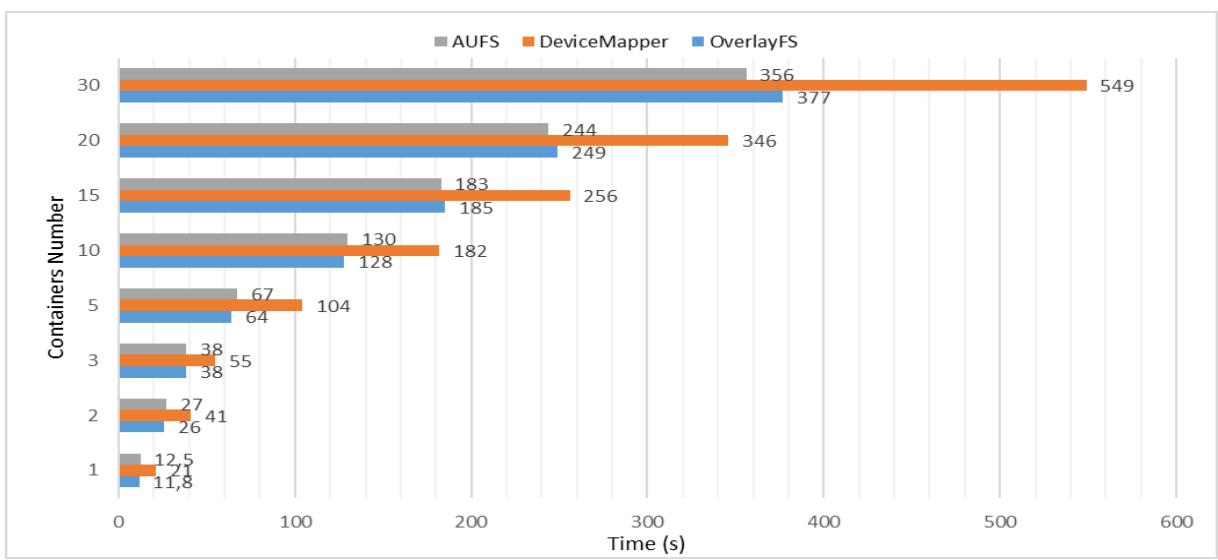


Figure 6. SVC Execution Time over Multiple Containers

application has not been considered in the reported results because not relevant for runtime performance analysis. Anyway, we have experienced an image build time in the order of a few minutes even on RaspberryPis, which will be rarely exploited as the nodes where the image building is performed; the reasonable process in a realistic production environment will be to delegate build operations to cloud resources and successively to migrate the ready images towards the interested fog nodes.

By coming back to the results of Table 1, as expected, the execution of native code outperforms the container-based one. This has demonstrated to be mainly due to the more efficient usage of file allocation table (fat) disk filesystem, of course in addition to the time for container creation. Among the case of Docker-based execution, the measured performance has shown to have a strong dependence on the filesystem used: Device mapper demonstrated to have the worst performance in this SVC application, mainly because of the high number of write operations it performs on the sparse file, with the introduction of a significant delay also in this case of single container execution; AUFS and OverlayFS have achieved comparable results on every operation, with a slightly better behavior of OverlayFS.

Finally, in Figure 6 we report the results about the concurrent execution of multiple containers over a single fog node. Also in this case, AUFS and OverlayFS outperform Device mapper, with a growing performance gap that has shown to be proportional to the number of containers in execution, thus demonstrating the practical unsuitability of using Device mapper over resource-constrained fog nodes in the relevant case of multiple concurrent containers scenario. Most relevant, Figure 6 clearly shows that, in terms of scalability while growing the number of running containers, also a resource-constrained IoT gateway such as a RaspberryPi node can achieve very good performance (linear dependency of execution time on the number of concurrently running containers), thus demonstrating the practical feasibility of the proposed approach. Let us finally and rapidly note that Figure 6 considers typical numbers of containers for deployment scenarios with fog nodes running on top of limited resource RaspberryPi devices. In fact, by considering a common fog deployment environment, hosting 20-30 concurrent containers is a worst-case scenario that widely covers very high-traffic and even unlikely (e.g., faulty or reliability-critical situations).

## 5. CONCLUSIVE REMARKS AND ONGOING RESEARCH WORK

Fog computing has established as a very powerful paradigm for large-scale IoT environments, having the capability to move part of the typical heavy-weight cloud computation closer to the endpoints, with significant advantages first of all in terms of minimized latency for control feedback and of reduction of global network traffic. In this paper, we originally address how to build a real fog middleware support by significantly extending industry-mature IoT gateways along two directions: i) decentralizing the MQTT broker functionality of the Kura framework from the cloud to the involved edges; and ii) exploiting industry-mature containerization to facilitate interoperability and portability via node configuration standardization in terms of macro-services and skeletons definition.

The good preliminary results described in this paper, first of all the good scalability of multiple container execution also over very resource-constrained nodes such as RaspberryPis, are encouraging

further research activities in the field. In particular, we are mainly working on designing original allocation algorithms for our Docker Swarm-based orchestrator in order to exploit global visibility of resource status and of available fog nodes, by trading off the greediness of our monitoring process and the acceptable distance from optimality of the allocation decisions taken. In addition, we are experimenting our fog computing middleware over a real application domain and a large-scale deployment environment where final devices are thousands of geographically distributed vending machines coordinated by either fixed or mobile IoT gateway fog nodes.

## REFERENCES

- [1] A. Faisal, D. Petriu, M. Woodside, "Network Latency Impact on Performance of Software Deployed Across Multiple Clouds", Int. Conf. Center for Advanced Studies on Collaborative Research (CASCON), 2013.
- [2] W. Wang, K. Lee, D. Murray, "Integrating Sensors with the Cloud using Dynamic Proxies", IEEE Int. Symp. Personal Indoor and Mobile Radio Communications(PIMRC), 2012.
- [3] N.D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, A.T. Campbell, "A Survey of Mobile Phone Sensing", IEEE Communications Magazine, Vol. 48, No. 9, pp. 140-150, Sept. 2010.
- [4] P. Bellavista, A. Corradi, C. Giannelli, "Mobile Proxies for Proactive Buffering in Wireless Internet Multimedia Streaming", 25<sup>th</sup> IEEE Int. Conf. on Distributed Computing Systems Workshops, 2005.
- [5] A. Toninelli, et al., "Supporting Context Awareness in Smart Environments: a Scalable Approach to Information Interoperability", Int. Workshop on Middleware for Pervasive Mobile and Embedded Computing. ACM, 2009.
- [6] P. Bellavista, A. Corradi, E. Magistretti. "REDMAN: An optimistic replication middleware for read-only resources in dense MANETs", Elsevier Pervasive and Mobile Computing, Vol. 1, No. 3, pp. 279-310, 2005.
- [7] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, "Fog Computing and Its Role in the Internet of Things", Int. Workshop on Mobile Cloud Computing (MCC), 2012.
- [8] F. Bonomi, R. Milito, P. Natarajan, J. Zhu, "Fog Computing: a Platform for Internet of Things and Analytics", Big Data and Internet of Things: A Roadmap for Smart Environments, Springer LNCI, Vol. 546, pp. 169-186, 2014.
- [9] R. Al Ali, I. Gerostathopoulos, I. Gonzalez-Herrera, A. Juan-Verdejo, M. Kit, M. Kit, B. Surajbali, "An Architecture-based Approach for Compute-Intensive Pervasive Systems in Dynamic Environments", 5th ACM/SPEC Int. Conf. on Performance Engineering (ICPE), 2014.
- [10] M. Firdhous, O. Ghazali, S. Hassan, "Fog Computing: Will it be the Future of Cloud Computing?", 3<sup>rd</sup> Int. Conf. on Informatics & Applications, Malaysia, 2014.
- [11] R. Bifulco, M. Brunner, R. Canonico, P. Hasselmeyer, F. Mir, F. Mir, "Scalability of a Mobile Cloud Management System", Int. Workshop on Mobile Cloud Computing (MCC), 2012.
- [12] P. Bellavista, A. Corradi, Alessandro Zanni, "Integrating Mobile Internet of Things and Cloud Computing towards Scalability: Lessons Learned from Existing Fog Computing Architectures and Solutions", 3rd Int. IBM Cloud Academy Conf. (ICA CON), 2015.
- [13] Kura. Available online at: <https://eclipse.org/kura>.

- [14] Docker. Available online at: <https://www.docker.io>.
- [15] W. Felter, A. Ferreira, R. Rajamony, J. Rubio, “An Updated Performance Comparison of Virtual Machines and Linux Containers”, Technical Report RC25482 (AUS1407-001), IBM Research Division, 2014.
- [16] A.M. Joy, “Performance Comparison between Linux Containers and Virtual Machines”, Int. Conf. Advances in Computer Engineering and Applications (ICACEA), 2015.
- [17] MQTT - Message Queue Telemetry Transport. Available online at: <http://mqtt.org>.
- [18] Mosquitto. Available online at: <http://mosquitto.org>.
- [19] O. Ferrer-Roca, R. Milito, “Big and Small Data: the Fog”, Int. Conf. on Identification, Information and Knowledge in the Internet of Things, 2014.
- [20] D. Bernstein, “Containers and Cloud: From LXC to Docker to Kubernetes”, IEEE Cloud Computing, Vol. 1, No. 3, pp. 81-84, 2014.
- [21] C. Pahl, B. Lee, “Containers and Clusters for Edge Cloud Architectures – a Technology Review”, 3<sup>rd</sup> Int. Conf. on Future Internet of Things and Cloud (FiCloud), 2015.
- [22] LXC - Linux Containers. Available online at: <https://linuxcontainers.org>.
- [23] CRIU - Checkpoint/Restore in Userspace. Available at: <http://www.criu.org>.
- [24] Docker Swarm. Available online at: <https://www.docker.com/products/docker-swarm>.
- [25] Kubernetes. Available online at: <http://kubernetes.io>.
- [26] Apache Mesos. Available online at: <https://mesos.apache.org>.
- [27] AUFS - Advanced multi layered unification filesystem. Available online at: <http://aufs.sourceforge.net>.
- [28] D. Merkel, “Docker: Lightweight Linux Containers for Consistent Development and Deployment”, Linux Journal, Vol. 239, 2014.
- [29] Device-mapper. Available online at: <http://www.sourceware.org/dm>.
- [30] OverlayFS. Available online at: <https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/Documentation/filesystems/overlayfs.txt>.

# Topology-aware resource management for HPC applications

Yiannis Georgiou

ATOS/Bull

Grenoble, France

[yiannis.georgiou@atos.net](mailto:yiannis.georgiou@atos.net)

Emmanuel Jeannot

Inria Bordeaux Sud-Ouest

Talence, France

[emmanuel.jeannot@inria.fr](mailto:emmanuel.jeannot@inria.fr)

Guillaume Mercier

Bordeaux INP

Talence, France

[mercier@labri.fr](mailto:mercier@labri.fr)

Adèle Villiermet

Inria Bordeaux Sud-Ouest

Talence, France

[adele.villiermet@inria.fr](mailto:adele.villiermet@inria.fr)

## ABSTRACT

The Resource and Job Management System (RJMS) is a crucial system software part of the HPC stack. It is responsible for efficiently delivering computing power to applications in supercomputing environments. Its main intelligence relies on resource selection techniques to find the most adapted resources to schedule the users' jobs. Improper resource selection operations may lead to poor performance executions and global system utilization along with an increase of the system fragmentation and jobs starvation. These phenomena play a role in the increase of the platforms' total cost of ownership and should be minimized. This paper introduces a new method that takes into account the topology of the machine and the application characteristics to determine the best choice among the available nodes of the platform based upon their position within the network and taking into account the applications communication pattern. To validate our approach, we integrate this algorithm as a plugin for SLURM, a popular and widespread HPC resource and job management system (RJMS). We assess our plugin with different optimization schemes by comparing with the default topology-aware SLURM algorithm using both emulation and simulation of a large-scale platform, and by carrying out experiments in a real cluster. We show that transparently taking into account the job communication pattern and the topology allows for relevant performance gains.

## CCS Concepts

•Information systems → Computing platforms;

## Keywords

resource management; job allocation; topology-aware placement; scheduling; SLURM

Publication rights licensed to ACM. ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of a national government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only.

ICDCN '17, January 04 - 07, 2017, Hyderabad, India

ACM ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007768>

## 1. INTRODUCTION

Computer science is more than ever a cornerstone of scientific development, as more and more scientific fields resort to simulations in order to help refine the theories or conduct experiments that cannot be carried out in reality because their scale or their cost are prohibitive. Currently, such computing power can be delivered only by parallel architectures. Larger and larger machines are being built around the world, and being able to display such a machine has become a challenge for states and nations, scientifically as well as politically.

However, harnessing the power of a large parallel computer is no easy task, due to several factors. First, this type of computer features usually a huge amount of computing nodes and this scale has to be taken into account when developing applications. Then, the nodes architecture has become more and more complex, as the number of cores per node is in constant increase from one generation of CPU to the next. The memory hierarchy becomes also more complex, as various levels of cache are now available and the rise of MCDRAM or NVRAM will make things even more complicated in the future. Indeed, an efficient exploitation of all these types of memories is possible only if the application developer takes it into account.

One way of dealing with such complexity would be to consider the application behavior (e.g. its communication pattern, or its memory accesses pattern) and to deploy it on the computer accordingly. To this end, the most widespread technique is to determine the list of cores on which the application has to be run on, and then to bind the processes on these cores so as to minimize/maximize a predetermined criterion (a.k.a. a metric). Such a technique has already been used and investigated to improve the performance of parallel applications [12].

However, a large parallel machine is often shared by many users running their applications concurrently. In such a case, an application execution will depend on its nodes allocation, as determined by the Resource and Job Management System (RJMS). Most of the time RJMS work in a best-effort fashion, which can lead to suboptimal allocations. That is, such allocations might be able to fulfill an application requirements in sheer terms of resources (number of CPUs, amount of memory) but might also fail to provide an environment tailored for an optimized execution. For instance, if the application processes communicate a lot between them-

selves, a set of nodes physically allocated apart from the rest might degrade performance severely. Furthermore, even if the given allocation is contiguous, taking into account process affinity leads to even better performance.

As a consequence, our goal is to apply to resource management the same technique that has proved its efficiency for applications deployment and execution, that is, taking into account an application’s behaviour in the process of reserving and allocating the needed resources (computing nodes). This means more criteria to be used and considered by the RJMS when a user submits its request to the system. Actually, taking in account an application behaviour when allocating nodes pushes even further the idea of using an application information to improve its execution.

In this paper, we shall detail the improvements we made to an existing RJMS in order to enable it to select the most suitable set of nodes for a given parallel application. To this end, we extend our TREEMATCH algorithm and integrate it in the SLURM software to improve its ability to match the resources to the actual application communication pattern. This paper is organized as follows: Section 2 gives an overview of the context and background of this work. Section 3 introduces all the software leveraged by this work before giving more technical insights about our topology-aware job allocation policy. Then Section 4 shows and discusses the results obtained while related work are listed in Section 5. Finally, Section 6 concludes this paper.

## 2. ISSUES OF RESOURCE ALLOCATION IN PARALLEL COMPUTERS

### 2.1 The Sharing of Resources

A large parallel computer is to some extent a tool that has to be exploited and used. A reason as why these computers increase in size and scale stems from the fact that some applications grow accordingly. Therefore, an adequate platform has to match these needs. However, a substantial part of the time, this large platform not only works in a time-sharing mode, but also in a space-sharing mode. Indeed, in order to exploit the hardware in a satisfactory way, several users share it, leading to a potentially very large number of users. An interactive access is therefore out of the question. To this end, the users have to submit their requests in terms of resources to a system called the Resource and Job Manager System (sometimes called a Batch Scheduler for short). This system’s goals are threefold: 1. to centralize and analyze all the received requests, 2. to allocate the most relevant type of resources (CPU, memory or network switches for instance) able to fulfill these demands and 3. to execute the application (a.k.a. the job) submitted by a user on the set of selected resources.

There are many and sometimes conflicting criteria that should be optimized by the RJMS. Then the question that pertains to this selection and allocation of resources is to choose one. For instance, one metric could be the system throughput, that is, the amount of jobs executed during a defined time step, whilst another could be the use (CPU load) of the system. All these metrics are relevant and which to use/optimize depends on a given point of view. That is, an administrator’s point of view might diverge from a user’s point of view. Indeed the users hardly possess a global view of the system (in most of cases) as opposed to the

administrators, hence the discrepancy.

### 2.2 Finding an Optimization Criterion

In this work, we focus on a metric relevant for users: the flow time (or turnaround time) that is, the time his/her job remains in the system. We believe it to be the most appealing one for a user seeking to gather results and get the outcome of his/her application as soon as possible. The question that now arises is how to speed up an application execution? Let us suppose that the developer has already optimized his/her application as much as it is possible. What are the means left to even speed things further up? One answer lies in the ecosystem of the application, that is, in the way the application is deployed and executed. In a previous work, unrelated to resource management and job scheduling, we showed that by taking into account an application behaviour when deploying it on the various processing entities (CPUs, cores, threads, etc.), it is possible to improve its global execution time [16, 13]. Actually, our goal is to improve the way an application accesses its data. This data locality can be improved in several ways, but we chose so far to use the communication pattern of the application, that is, an expression of the amount of bytes/messages exchanged by the application processes. Then, we try to match this pattern to the underlying architecture by following the principle that the more processes are communicating with the others, the closer the cores they should be bound to. This can be done by several techniques but usually involves process binding and rank reordering [17].

However, the execution still depends on the set of resources allocated to the application by the RJMS. Since no guarantee is given that this allocation will be compliant with the application communication pattern, some negative side effects may occur. For instance, a subset of nodes might be physically far from another subset, thus impacting the communication between processes belonging to each subset. As a consequence, an allocation that takes into account an application communication scheme leads to performance improvements. To that end, we consider a well-known and widespread RJMS called SLURM and design a new plugin based on the TREEMATCH algorithm. So far, TREEMATCH was used to compute a matching between the application processes and the physical cores available. Now, we use it to determine a nodes allocation before deploying the application.

Hence, to improve the flow time we aim at reducing the job execution time of the submitted application by improving its mapping.

### 2.3 A Motivating Example

The goal of this work is to apply mapping techniques (e.g. TREEMATCH) before the execution of the application processes and compare different approaches. We assume that the communication pattern of the application is known at submission time. Such a communication pattern can be gathered with application monitoring (see Section 3.1.2) or by analyzing the structure of the parallel algorithm (for instance if we are dealing with a stencil code we know which processes are communicating together and the amount of exchanged data). In any case, we assume that this communication pattern remains unchanged from one run to the other. It is not the case for all parallel applications but a large amount of applications comply to these models (for instance,

Proc.	0 - 1	2 - 3	4 - 5	6 - 7
0-1	0	<b>20</b>	0	<b>2000</b>
2-3	<b>20</b>	0	<b>1000</b>	0
4-5	0	<b>1000</b>	0	<b>10</b>
6-7	<b>2000</b>	0	<b>10</b>	0

Table 1: Affinity matrix for 8 processes (4 groups of 2 processes each). Shows the amount of bytes/messages exchanged by the application processes

dense linear applications and kernels, stencil codes, regular mesh partitioning based applications, etc.). When the communication pattern changes from one run to an other, the proposed solution is not applicable and the user has to fall back to a standard allocation scheme: mixing the proposed topology-aware mapping with other types of mapping is totally acceptable.

Several possibilities are available. The most obvious one is to not use TREEMATCH at all and let the SLURM environment deal with the topology by itself. The second possibility is to apply TREEMATCH just before the job execution, once SLURM has selected the resources. Another possibility is to use TREEMATCH inside the selection mechanism of SLURM.

An example of the difference between these approaches is depicted by Fig. 1. Let us suppose that we have 6 nodes composed of two computing entities each. We assume that node n3 is not available as computing entities 6 and 7 are already used by an other application, hence unavailable for a job allocation. Let us assume that a newly submitted job requests 4 nodes. For the sake of simplicity, we group processes in pairs (0-1, 2-3, etc.) and hence each pair of processes shall be assigned to one node. The affinity matrix is given in table 1.

If SLURM has to allocate resources for these 8 processes, it will look for the smallest number of switches able to fulfill the request. In this case, it will require to use the whole tree. Then, it will allocate processes from left to right inside nodes in a round-robin fashion. It will allocate nodes 0, 1, 2 and 4 for the job and then map processes onto the computing entities. We can see that such an allocation is rather costly communication-wise as groups of processes are spread onto the entities and no optimization is enforced in this regard. It is therefore possible to call TREEMATCH (see Section 3.1.3) to optimize the process mapping on these entities accordingly to the affinity matrix. By doing so, the resulting mapping is: group 0-1 on n0, group 6-7 on n1, group 2-3 on n2 and 4-5 on n4. This is the best possible solution once the resources have been allocated. However, group 2-3 communicates a lot with group 4-5. With such an allocation, all the communications will transit through the root of the topology, a costly solution in terms of hops. However, a better outcome is achievable if TREEMATCH performs the resource allocation. Given such a topology and the above affinity matrix, TREEMATCH will allocate group 0-1 on n0, group 6-7 on n1, group 2-3 on n4 and 4-5 on n5 since there are constraints on node n3.

In this case, all the communication between group 2-3 and group 4-5 will take only 2 hops instead of 4 and therefore the communication cost is even more reduced.

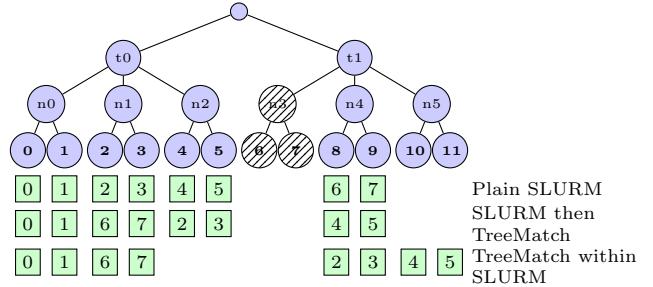


Figure 1: Tree topology of 6 nodes of 2 processing units with one unavailable node: n3

### 3. A TOPOLOGY-AWARE RESOURCE AND JOB MANAGEMENT SYSTEM

#### 3.1 Software

In this section, we introduce with more details the various software elements that we use to implement the work described in this paper. First, we shall describe SLURM, our target RJMS. Then, we shall explain the method employed to gather information about the application communication scheme (a.k.a. our *affinity matrix*). Then, we give more specific information about the TREEMATCH algorithm and the constraints mapping extension we have implemented.

##### 3.1.1 SLURM

We implement a new topology-aware placement algorithm within the open-source resource and job management system SLURM [28]. SLURM performs workload management on six of the ten most powerful computers in the world of the Top500 list<sup>1</sup> including the system ranked number one, Tianhe-2, which features 3,120,000 computing cores.

SLURM is specifically designed for the scalability requirements of state-of-the-art supercomputers. It is based upon a centralized server daemon, `slurmctld` also known as the controller, which communicates with client daemons `slurmd` running on each computing node. Users can request the controller for resources to execute interactive or batch applications, referred to as jobs. The controller dispatches the jobs on the available resources, whether full nodes or partial nodes, according to a configurable set of rules. The SLURM controller also features a modular architecture composed of plugins responsible for different actions and tasks such as: job prioritization, resources selection, task placement or accounting.

The resource selection process within SLURM takes place as part of the global job scheduling procedure. In particular, this procedure makes use of the `plugin/select`, which is responsible for allocating the computing resources to the jobs. Other plugins are used to facilitate and extend this procedure such as `plugin/topology` which takes into account the network topology of the cluster, the `plugin/gres` which can extend the allocation to different generic resources and the `plugin/task` which provides the isolation and possible binding of tasks on the resources.

There are various resource selection plugins within SLURM that can take into account the specificities of the underlying platforms' architecture such as `linear` and `cons_res`.

<sup>1</sup><http://top500.org/lists/2015/11/>

The **select/linear** plugin allows the allocation of complete nodes for jobs, using simple and scalable best-fit algorithms, however, the lower granularity of allocatable unit is the node which is quite limiting for new multicore and manycore architectures. The **select/cons\_res** plugin is ideal for this type of architectures where nodes are viewed as collections of consumable resources (such as cores and memory). In this plugin, nodes can be used exclusively or in a shared mode where a job may allocate its own resources different than other jobs using the same node. The algorithms within the **cons\_res** plugin are also scalable, featuring best-fit placement of jobs but they are more complex than **select/linear** since a finer granularity of allocatable resources is taken into account. One of the first version of the **select/cons\_res** plugin is described in [2].

Our studies and developments as described in the following sections are based upon the **select/cons\_res** plugin therefore we try to analyze a bit more some important internals of this plugin. The internal representation of resources and availabilities within SLURM is made using bitmap data structures. In the case of the linear plugin only a node bitmap is needed whereas in the case of the **cons\_res** plugin, besides the node bitmap, a core bitmap is used to represent internal node resources availabilities. Within the **cons\_res** plugin, the usage of node and core bitmaps is leveraged efficiently (e.g. kept separated in different contexts) in order to keep a high scalability for the selection algorithms. Another functionality of the **cons\_res** plugin is the distribution of tasks within the allocated resources, which is an important feature for the optimal performance of parallel applications.

SLURM provides configuration options to make the resources selection network topology-aware through the activation of the topology plugin (**topology/tree** plugin). A particular file describing the network topology is needed and the job placement algorithms favor the choice of groups of nodes that are connected under the same network switch. The goal of the SLURM topology-aware placement algorithms is to minimize the number of switches used for the job and provide a best-fit selection of resources based on the network design. This feature becomes mandatory in the case of pruned butterfly networks where no direct communication exists between all the nodes. We use this plugin in our experiments. The scalability and efficiency of topology-aware resource selection of SLURM has been evaluated in [10].

Finally since the **cons\_res** plugin deals with multi-core architectures the isolation and binding of tasks upon the used resources is an important feature to guarantee a minimal interference between jobs sharing nodes. This feature takes place through the usage of the **task/affinity** or the **task/cgroup** plugin which use linux kernel mechanisms such as cgroups and cpusets or APIs such as hwloc [5] in order to provide the described isolation and binding.

### 3.1.2 Application Monitoring

For this work we need to model an application communication scheme. The way communications occur describes the affinity between processes. For the affinity matrix, we gather the communication pattern thanks to a dynamic monitoring component we integrate in Open MPI as an MCA (Modular Component Architecture) framework called pml (point-to-point management layer). This component, when activated at launch time, monitors all the communications at the lowest level in the Open MPI stack (i.e. once collec-

tive communications have been decomposed into point-to-point operations). Therefore, as opposed to the standard MPI profiling interface (**PMPI**) approach where the MPI calls are intercepted, we monitor in our case the actual point-to-point communications that are issued by Open MPI, which is much more precise: for instance, we can see the tree used for aggregating values in a **MPI\_Gather** call.

Internally, this component uses the low-level process ids and creates an associative array to convert sender and receiver ids into ranks in **MPI\_COMM\_WORLD**. At the end of the execution, each process dumps its local view into a file and a script aggregates all the local views at a given process to get the full communication matrix.

### 3.1.3 TreeMatch

TREEMATCH [13] [12], is a library for performing process placement based on the topology of the machine and the communication pattern of the application, for multicore, shared memory machines as well as distributed memory machines. It computes a permutation of the processes to the processors/cores in order to minimize the communication cost of the application.

To be more specific, it takes as input a tree topology (where the leaves stand for computing resources and internal nodes correspond to switches or cache levels) and a matrix describing the graph affinity between processes. The topology information is supplied either by the RJMS or by tools such as hwloc or netloc<sup>2</sup>. A hierarchy is extracted from this graph so that it matches the hierarchy of the topology tree. The outcome is a mapping of the processes onto the computing resources. The objective function optimized by TREEMATCH is the Hop-Byte [29], that is, the number of hops weighted by the communication cost:

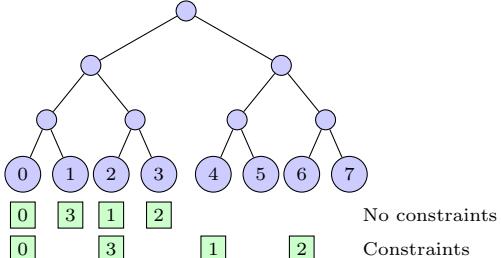
$$\text{Hop-Byte}(\sigma) = \sum_{1 \leq i < j \leq n} \omega(i, j) \times d(\sigma(i), \sigma(j))$$

where  $n$  is the number of processes to map,  $\sigma$  is the process permutation output produced by TREEMATCH (process  $i$  is mapped on computing resource  $\sigma(i)$ ),  $A = (\omega_{i,j})$   $1 \leq i \leq n$ ,  $1 \leq j \leq n$  is the affinity matrix between these entities and hence  $\omega(i, j)$  is the amount of data exchanged between process  $i$  and process  $j$  and  $d(p_1, p_2)$  is the distance, in number of hops, between computing resources  $p_1$  and  $p_2$ . In a previous work [13], we have shown that minimizing this metric allows for application runtime reduction for tree-based topologies.

An important feature of TREEMATCH is that it only uses the structure of the tree and does not require a precise valuation of the speed of the links in the topology. Therefore, TREEMATCH does not require a performance assessment of the system on which the application is going to be executed. We believe this to be a strong advantage, as gathering such information is error-prone, might be incomplete and subject to inaccuracy.

In order to tackle the fact that not all resources are available for mapping we enhance TREEMATCH from [13] to take constraints into account. When not all leaves are available for mapping (because some of them are already used by other applications), it is possible to restrict the leaves onto which processes can be mapped such that only a subset of the nodes is used for the mapping. To do so, we use a recursive k-

<sup>2</sup><https://www.open-mpi.org/projects/netloc>



(a) Input tree topology example.

Proc.	0	1	2	3
0	0	5	10	100
1	5	0	20	5
2	10	20	0	10
3	100	5	10	0

(b) Affinity matrix example.

Figure 2: Example of TreeMatch output (green square) based on the affinity matrix and the tree topology. The first line is without constraints: in this case the hop-byte metric is 360. The second line is when only cores with even numbers are allowed to execute processes (hop-bye is 660 in this case)

partitioning algorithm where we add dummy processes that are forced to be mapped onto unavailable resources while real processes are mapped to actual available resources.

In Fig. 2, we describe an example where we map 4 processes on an architecture featuring 8 computing resources and structured as a 3-levels tree. We display 2 cases: one without constraints and the other where only cores with even numbers are available for mapping.

### 3.2 Job Allocation Strategy

We implement a new selection option for the SLURM `cons_res` plugin. In this case the regular best-fit algorithm used for nodes selection is replaced by TREEMATCH.

To this end we need to provide three pieces of information: a job affinity matrix, the cluster topology and the constraints due to other jobs allocations.

The communication matrix is provided at job submission time through a distribution option available in the `srun` command:

```
srun -m TREEMATCH=/comm/matrix/path cmd
#SBATCH -m TREEMATCH=/comm/matrix/path.
```

Its location (path) is then stored by the SLURM controller in the data structure describing a job and can be used by TREEMATCH for allocation.

As for the global cluster topology, it is provided to the controller by a new parameter in the configuration file: `TreematchTopologyFile=/topology/file/path`.

Whenever a job allocation is computed, this topology is completed by constraints informations. These constraints are provided by the nodes and cores bitmaps used by the SLURM controller to describe the cluster utilization. We need to translate this topology description into the TREEMATCH topology.

TREEMATCH considers computing units as selection granularity and assign them an id considering the global topology. It must be the same for the SLURM selection plugin using TREEMATCH. Hence we use the `cons_res` plugin

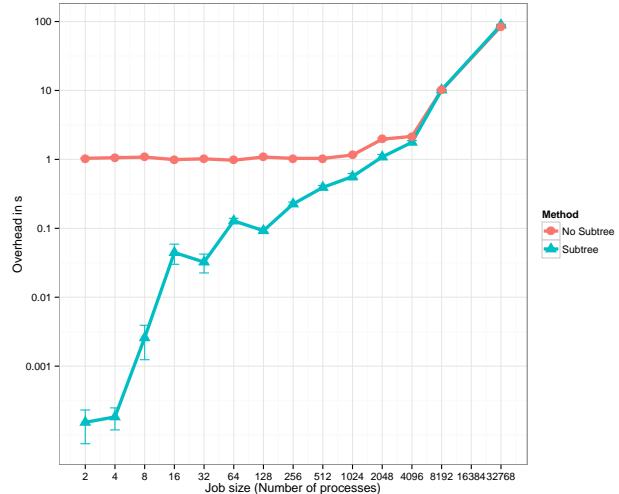


Figure 3: Comparison of TREEMATCH overhead (s) for different job size on a cluster with 80640 cores between methods with and without subtrees

gin with the configuration `SelectTypeParameters=CR_CPU` or `CR_Core`. In this case SLURM uses a cores bitmap describing precisely the location of unused CPUs inside nodes relatively to the nodes bitmap. Therefore, we need to translate SLURM local CPU ids into global TREEMATCH CPU ids. Then, we use the constraints feature of TREEMATCH (described in Section 3.1) to only use CPUs not already allocated to a running job. The CPUs chosen by TREEMATCH must then be translated again in new bitmaps for SLURM to use.

However, in the case of a large topology, our algorithm overhead increases: the larger the topology, the longer the TREEMATCH algorithm takes. To reduce this time, we also implement an alternative method which first finds a subtree in the global topology. Then, TREEMATCH uses this subtree to rapidly choose the job allocation. To find this subtree we search through the topology tree from the leaves up to the root and from left to right. We stop as soon as we find a node with enough unused CPUs. For instance, if we consider Fig. 1 and we assume that node n0 is occupied instead of n3, then the first tree with 2 CPUs is n1 and if we need 6 CPUs, we shall select subtree t1.

Fig. 3 compares the overhead of this algorithm with and without subtree utilization on a cluster featuring 80640 cores. It shows that, for jobs using less than 4096 cores, the subtree technique reduces the overhead. In any case both approaches takes less than 1s. At some point, the time increases linearly with the application size. However, as shown in the experiments (Sec. 4), the TREEMATCH overhead is largely compensated by the execution time gain. Moreover, for large applications, it is possible to compute the mapping at the node level (instead of computing it at the core level): hence a full-size application (80640 cores) requires 5040 nodes which leads to an overhead of a few seconds.

For the experiments described in Section 4 we need to modify the jobs run times dynamically according to their allocation. To do this we compute for each job both the SLURM allocation and the TREEMATCH one. Then we compute  $R$ , the ratio between their hop-byte cost (c.f. Section 3.1). We

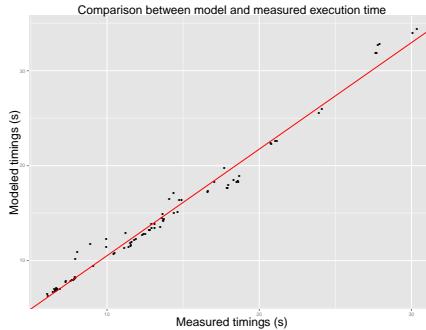


Figure 4: TREEMATCH measured time vs. modeled time for the minighost application with a communication ratio between 5% and 45%

model job runtimes with computation times and communication times:  $T = T_{calc} + T_{comm}$ . Let  $\alpha$  be the ratio of communication time of the whole runtime:  $T_{comm} = \alpha T$ . Hence,  $T = \alpha T + (1 - \alpha)T$ . TREEMATCH impacts only the communication cost. Therefore, we model the execution time  $T'$  using the TREEMATCH allocation with:

$$\begin{aligned} T' &= T_{calc} + RT_{comm} = R\alpha T + (1 - \alpha)T \\ &= (1 + R\alpha - \alpha)T \end{aligned}$$

We validate this model with the minighost application [3] that computes a stencil in various dimensions. We execute 84 runs with various settings (number of processors, different parameters) using a round-robin placement or a mapping computed with TREEMATCH. The minighost output also provides the percentage of communication in a run. In our case, this ranges from 5% to 45%. Fig. 4 shows the validation of the above model. On the x-axis is the TREEMATCH runtime and on the y-axis is the predicted time based on the ratio  $R$  of the hop-byte of the TREEMATCH mapping and the SLURM mapping,  $\alpha$  the percentage of communication and  $T$  the measured execution runtime. We see a very strong correlation between both timings even though the modeled timings tend to be slightly larger than the real ones.

## 4. EXPERIMENTAL VALIDATION

### 4.1 Emulation Experimental Setup

Our experiments have been carried out on the Edel cluster from the Grid'5000 Grenoble site. Edel is composed of 72 nodes featuring 2 Intel Xeon E5520 CPUs (2.27 GHz, 4 cores/CPU) and 24GB of memory.

We emulate Curie (a TGCC cluster with 5040 nodes and 80640 cores<sup>3</sup>) using a SLURM internal emulation technique called `multiple-slurmd` initially described and used in [10]. SLURM uses deamons: one `slurmctld` as the controller and one `slurmd` on each node. To emulate a larger cluster, we use 16 Edel nodes and launch 315 `slurmd` daemons on each node. We can consequently submit jobs as if we were working on the Curie cluster, emulating all the job scheduling overheads. We use simple jobs (just performing a call to `sleep`) in order to provide the necessary time and space illusion to the controller that a real job is actually executing.

We base our experiments on a Curie workload trace taken from the Parallel Workload Archive<sup>4</sup>. We have two sets of jobs. The first one is to fill the cluster, and the jobs belonging to this set are always scheduled using SLURM in order to have the same starting point for all the experiments. The second set, called the *workload*, is the one we actually use to compare the different strategies.

All the measurements are done through the SLURM loggin system which gives us workload traces similar to the ones we obtain from Curie.

Finally, to use TREEMATCH we need to provide each job with a communication matrix. For these experiments we use randomly generated matrices featuring various sparsity rates. Indeed, the name of the application does not appear in the workload trace and therefore we cannot know the gain an optimized mapping would yield for a given entry in the trace. However, this depends on the communication ratio of the application (the higher this ratio, the larger is the possible gain in terms of runtime) and its communication pattern. Here, based on recent results [8, 11], we design the matrices such that the gain is similar to real-world applications. On average, the observed gain is 4% (resp. 11% and 18%) for a communication ratio of 10% (resp. 30% and 50%).

### 4.2 Emulation Results

We compare 4 cases : the classical topology-aware SLURM selection (SLURM), the same but using TREEMATCH for process placement after the allocation process and just before the execution starts (TM-A), TREEMATCH used both for the allocation process and for the process placement (TM-I) and finally the same but using the subtree technique to reduce the overhead (TM-Isub).

To evaluate our results, we use several metrics (two are for the whole workload and two are for each individual job):

- makespan: this is the time taken between the submission of the first job and the completion of the last job of the *workload*.
- utilization: this is the ratio between the CPUs used and the total number of CPUs in the cluster during the execution of the *workload*.
- job flowtime (or turnaround time): this is the time between the submission and the completion of a given job.
- job runtime: this is the time between the start and the completion of a given job.

In our case, the *workload* comprises 60 jobs. To keep the duration reasonable we decrease the jobs runtimes by a 50% factor. Figure 5 describes the results obtained for this workload and two values of  $\alpha$  (1/3 and 1/2). Figure 5a shows that using TREEMATCH to reorder the process ranks reduces the makespan but using it inside SLURM to allocate nodes decreases it even more. This is what is shown in Fig. 1: incorporating TREEMATCH in SLURM gives more room for optimization as the mapping is not constraints by the allocation. Moreover, the subtree optimization leads to comparable results than without the optimization. This is due to the fact that in this case the makespan is determined by a small set of jobs and hence the impact of this optimization

<sup>3</sup><http://www-hpc.cea.fr/en/complexe/tgcc-curie.htm>

<sup>4</sup><http://www.cs.huji.ac.il/labs/parallel/workload/>

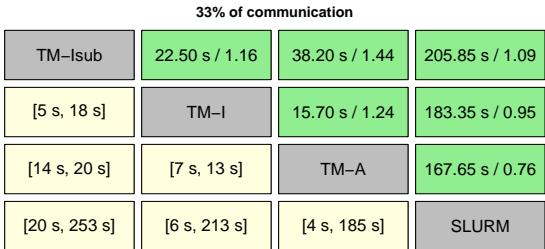
Com	SLURM	TM-A	TM-Isup	TM-I
50%	8318	6407	6073	6077
33%	8316	7502	6821	6887

(a) Makespan

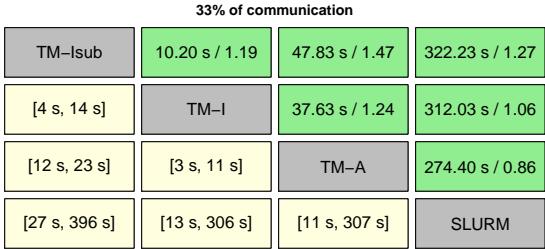
Com	SLURM	TM-A	TM-Isup	TM-I
50%	33%	42%	44%	44%
33%	33%	36%	40%	39%

(b) Utilization

Figure 5: Workload Metrics for the different strategies and different amount of communication ratio



(a) 33% of communication



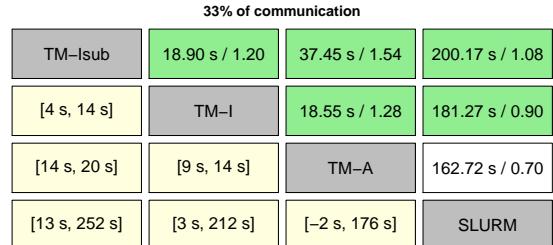
(b) 50% of communication

Figure 6: Statistical comparison of selection methods: flow time

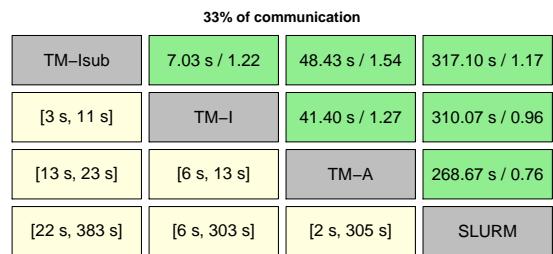
is not visible for this metric. We also see that the larger the communication ratio the greater the gain, this is expected as TREEMATCH optimizes communication only. This is tested through simulation in Section 4.3.

Figure 5b also shows that for the same submission workload, TREEMATCH improved the resource utilization.

In Fig. 6 and Fig. 7, we use paired comparisons between different strategies for respectively jobs flowtime and jobs runtime. Here, we consider job-wise metrics, therefore we want to understand if, when we average all the jobs, a strategy turns out to be better than another. Each strategy is displayed on the diagonal. On the upper right, we have the average difference between the strategy on the column and the one on the row and the geometric mean of the ratios. For instance, in Fig. 6a, we see that on average the job flowtime is 183.35s faster with TM-I than with SLURM and the average ratio is 0.95. On the lower left part, we plot the 90% confidence interval of the corresponding mean. The interpretation is the following: if the interval is positive, then the strategy on the row is better than the strategy on the line with a 90% confidence. In this case, the correspond-



(a) 33% of communication



(b) 50% of communication

Figure 7: Statistical comparison of selection methods: runtime

ing mean is highlighted in green. If the interval is negative the strategy on the line is better than the one on row and the corresponding mean is highlighted in red. Otherwise, we cannot statistically conclude with a 90% confidence on which strategy is the best and we do not highlight the corresponding mean. For example, on Figure 6a we can see that using TREEMATCH in SLURM is better than not using it. Moreover, here we see that using the subtree optimization improves the metric. For all the cases we see that TM-Isup is better than TM-I that is better than TM-A. Therefore, restricting the usage of TREEMATCH improves the performance as the gain in computing a solution overcome the loss in terms of quality of this solution.

Moreover, both flowtime and runtime using TREEMATCH in SLURM are shorter than using TREEMATCH after SLURM, with a ratio between 1.44 and 1.54. We can also see that the more an application communicates, the smaller are the average gaps. For example, between TM-I and TM-Isup (with a 33% of communication ratio), the average difference is 22.5 s, but for a 50% ratio it is 10.2s. In these experiments, the cluster is already full when submitting the first jobs. Therefore, a part of their flowtime corresponds to the wait for a free allocation.

Figure 7 shows the comparison of jobs runtimes. We observe similar behavior except that the confidence interval between SLURM and TM-A does not allow to conclude with 90% confidence that TM-A is better than SLURM.

Through these experiments we observe that using TREEMATCH in the allocation process induces no negative effects and improves the global use of a cluster. Moreover, from a user point of view, using TREEMATCH can also be profitable by decreasing the runtime of his/her jobs.

### 4.3 Simulation Results

As the experiments done in the above section are carried

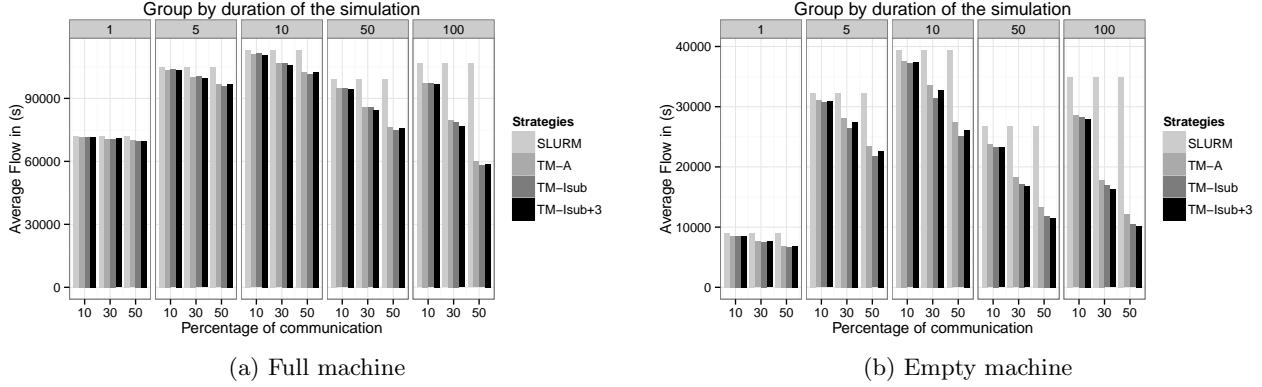


Figure 8: Average flowtime using the Curie trace with different strategies and various percentage of communication

out through emulation, they are very long to compute (as long as the real execution times). In order to cover a larger set of test cases and a longer time-scale we design a simulator that simulates both the job selection part and the job execution part. Our simulator is an event-based one that reads the machine topology and the job submission trace workload and computes the start time and end time of each job based on its duration (given by the workload) and the allocation. For the job selection step, we implement the same algorithm than we use in the above section and the time to compute the allocation is based on the duration of TREEMATCH when it is used or is set to 2 seconds when SLURM is used. For the execution time part, we use the formula shown in Section 4.1 if we use TREEMATCH. Otherwise, the duration given by the Curie trace is used. It is accurate enough to provide makespan duration with an average absolute error of less than 3% for Figure 5a.

The experiments described here represent more than 12 millions node-hours. Note that allocations above 1 million node-hours are only given to application projects proposal by supercomputing centers and never to software stack optimization projects<sup>5</sup>, hence justifying the use of simulation.

Figure 8a shows the average flow of the jobs in the case where the machine is already full of jobs (that have all been scheduled using the regular SLURM strategy). We group the measurements by simulation duration (i.e. for group 50, we consider only the jobs submitted during the first 50 hours): we go from 1 hour (365 jobs) to 100 hours (13687 jobs). On the x-axis we display the different percentage of communication (from 10% to 50%) and we have 4 strategies: the plain SLURM, TREEMATCH applied at the beginning of the job to map processes to resources after SLURM has allocated the nodes (TM-A), and TREEMATCH used in SLURM to compute the allocation and the mapping using the minimal subtree (TM-Isub), or 3 levels above the minimal subtree (TM-Isub+3). We see that the impact of TREEMATCH on the flow increases with the duration of the simulation because at the beginning of the simulation, as the machine is full, the flowtime depends mainly on the time a job has to wait before starting, while as time goes the impact on the improvement of the mapping due to TREEMATCH accumulates. Here, we do not see much difference between the

different strategies involving TREEMATCH because there is less room for optimization when the machine is fully utilized. However, the results are consistent, in terms of quality with the emulation results presented in the previous section.

Figure 8b shows the average flow of the jobs in the case where the machine is totally empty. In this case, we see that the gain with TREEMATCH increases and appears earlier which corroborates the hypothesis made in the previous paragraph. Moreover, as we have more opportunities for optimization we see that using TREEMATCH in SLURM is more beneficial than using it just before the job execution. We also see a large gap for hour 5 because in the workload a large job (32768 cores) is submitted at 8461s, that takes a long time to schedule and that uses a substantial part of the machine, thus impacting all the subsequent jobs.

As in the previous section, we see that even with a small average gain on each jobs (4% for the 10% communication ration case to 17% for the 50% communication ratio), we are able to achieve very large gain on the overall. This is due to the fact that these gains accumulates during the workload lifetime. We therefore expect even greater gains on real settings as the operational lifetime of a real machine is much longer than the experiments done here.

## 5. RELATED WORKS AND DISCUSSION

The idea of using the most adequate hardware resource to a specific application is not new and has been explored in previous work. It has been particularly popular in the context of grids environments ([15], [25], [23]) where it is important to select the best set of resources (clusters in this case) to use. Such work try to reduce the impact of WAN communication in grids but do not address the deeper details of the physical topology, such as NUMA effects or cache hierarchy for instance.

More recently, some works have targeted a specific type of applications, that is, MapReduce-based applications. For instance, the TARA [14] uses a description of the application to allocate the resources. However, this work is tailored for a very specific class of applications and does not address hardware details.

The mapping of a parallel applications' tasks to the physical processors based on the network topology can lead to important performance improvements [4]. Network topology characteristics can be taken into account by the scheduler [18] so as to favor the choice of group of nodes that are

<sup>5</sup>See PRACE core hours award for instance: <http://www.prace-ri.eu/hermit-awardees>

placed on the same network level, connected under the same network switch or even placed close to each other so as to avoid long distance communications. This kind of feature is taken into account by most of open-source and proprietary RJMSs. However even if most of them use the characteristics of the underlying physical topology, they eventually fail to take into consideration the application behaviour when allocating resources and this is something that this work specifically addresses. HTCondor (formerly Condor) leverages a so-called *matchmaking* approach [21] that allows it to match the applications needs to the available hardware resources. However, the application *behaviour* is not part of this matchmaking and HTCondor targets both clusters and networks of workstations. SLURM [28], as previously described, provides an option to minimize the number of network switches used in the allocation, so as to reduce the communication costs during the application execution (switches that are the deeper in the tree topology are supposed to be the less costly than upper ones). The same idea of topology-aware placement is exploited by PBS Pro [20], Grid Engine[19], and LSF [24]. Fujitsu [9] provides the same but only for its proprietary Tofu network. As far as our knowledge, SLURM [28] remains the only one providing a *best-fit* topology-aware selection whereas the others propose *first-fit* algorithms.

Some other RJMS offer task placement options that can enforce a clever placement of the application processes. That is the case of Torque [7] which proposes a NUMA-aware job task placement. OAR [6] uses a flexible hierarchical representation of resources which offers the possibility to place the application processes upon the hierarchy within the computing node. However, in these existing works, only the network topology is taken in account and the nodes internal architecture is left unaddressed when performance gains are expected from exploiting the memory hierarchy.

Jingjin Wu et al. in [26] introduced a hierarchical task mapping strategy for modern supercomputers based on generic recursive algorithms for both fat-tree and torus network topologies showing very good performance with low overhead. Rashti et al. [22] proposed a weighted graph model for the whole physical topology of the computing system, including both the inter and intra node topologies. Even if both previous related works have shown interesting results with application sets, they have not been integrated with real resource and job management system neither tested with real workload traces which is our case in this paper.

A study for torus network topology [1] showed how processor ordering takes place based on space filling curve, such as Hilbert Curve, to map the nodes of the torus onto a 1-dimensional list in order to preserve locality information. This paper described the study about the allocation strategies implemented on the proprietary Cray Application Level Placement Scheduler (ALPS). Similar strategies, have been recently incorporated within SLURM with<sup>6</sup> (or without<sup>7</sup>) the use of ALPS. Another interesting work [27] adapted only for torus topology, presented a window-based locality-aware job scheduling strategy that tries to optimize job and system performance in the same time. Its goal is to preserve node contiguity by considering multiple jobs for scheduling while making use of the 0-1 Multiple Knapsack problem for resource allocation. The last 2 related works do not con-

sider communication patterns as parameters within the algorithms.

Several binding policies are available, and they are compatible with the policies implemented in Open MPI. In all these solutions, the user has to retrieve the architectural details before submitting his/her job. Also, the placement options offered leave the user with the burden to determine his/her policy beforehand, and the application communication scheme is not taken into account.

In our case, we improve this functioning on three levels: first, we take into account not only the network but also the node internal structure. The information used is based on the *structure* of the nodes and the memory hierarchy. In other words, we do not use latency and bandwidth figures to compute our allocation. Then, this information is retrieved directly by our plugin does not have to be supplied by the user. All the technical details are hidden. Last, but not least, we also take into account not only the architecture but also the application behaviour both for the allocation and the execution of a job.

## 6. CONCLUSIONS

Job scheduling plays a crucial role in cluster administration, enabling both better response time and resource usage. In this paper, we tackle the problem of allocating and mapping jobs according to a cluster topology and application process affinity. We extend TREEMATCH to design a new allocation policy that allocates and maps at the same time application processes on the resources, based on the communication matrix of the considered application. Such strategy is implemented in the SLURM `cons_res` plugin. We test this strategy on emulation and simulation and compare it with the standard SLURM topology-aware policy and the method consisting in mapping processes after the allocation is determined.

Results show that taking into account application characteristics and the topology provides better makespan, flow time, utilization and job runtime compared to the standard topology-aware and compact SLURM policy. We also show that the level at which we consider the topology impacts the performance. It is better to have a more local view of the topology than only a global view since in this latter case, allocation quality is slightly better but longer to compute. Last, even if not all the jobs are able to use this strategy all of them benefit from it with a reduced flowtime.

For future work, we would like to investigate the following research axes. First, we would like to look at fragmentation metrics. Indeed, the way jobs are allocated impacts the global resource usage and this aspect should be quantified. Also, we would like to find means to gather in a systematic fashion applications communication patterns in order to create an applications classification based on these patterns and then implement this solution in production. We would also like to validate this approach in other job scheduler such as OAR [6]. Concerning SLURM integration and extensions, we are currently working on the inclusion of our new developments in the next official SLURM release.

## 7. ACKNOWLEDGMENTS

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER

<sup>6</sup>[http://slurm.schedmd.com/cray\\_alps.html](http://slurm.schedmd.com/cray_alps.html)

<sup>7</sup><http://slurm.schedmd.com/cray.html>

and several Universities as well as other organizations (see <https://www.grid5000.fr>). Part of this work is also supported by the ANR MOEBUS project ANR-13-INFR-0001. This work is partially funded under the ITEA3 COLOC project #13024.

## 8. REFERENCES

- [1] C. Albing, N. Troullier, S. Whalen, R. Olson, J. Glenski, H. Pritchard, and H. Mills. Scalable node allocation for improved performance in regular and anisotropic 3d torus supercomputers. In *EuroMPI 2011, Santorini, Greece*, pages 61–70, 2011.
- [2] S. M. Balle and D. J. Palermo. Enhancing an open source resource manager with multi-core/multi-threaded support. In *JSSPP 2007, Seattle, WA, USA*, pages 37–50, 2007.
- [3] R. F. Barrett, C. T. Vaughan, and M. A. Heroux. Minighost: a miniapp for exploring boundary exchange strategies using stencil computations in scientific parallel computing. *Sandia National Laboratories, Tech. Rep. SAND2011-5294832*, 2011.
- [4] A. Bhatele, E. J. Bohm, and L. V. Kalé. Topology aware task mapping techniques: an api and case study. In *PPOPP*, pages 301–302, 2009.
- [5] F. Broquedis, J. Clet-Ortega, S. Moreaud, N. Furmento, B. Goglin, G. Mercier, S. Thibault, and R. Namyst. Hwloc: a Generic Framework for Managing Hardware Affinities in HPC Applications. In *PDP 2010*, Pisa, Italia, Feb. 2010.
- [6] N. Capit, G. Da Costa, Y. Georgiou, G. Huard, C. Martin, G. Mounié, P. Neyron, and O. Richard. A batch scheduler with high level components. In *Cluster computing and Grid 2005 (CCGrid05)*, Cardiff, United Kingdom, 2005. IEEE.
- [7] A. computing. Torque resource manager. <http://docs.adaptivecomputing.com/torque/6-0-0/Content/topics/torque/2-jobs/monitoringJobs.htm>.
- [8] E. H. Cruz, M. Diener, L. L. Pilla, and P. O. Navaux. An efficient algorithm for communication-based task mapping. In *PDP'2015*, pages 207–214. IEEE, 2015.
- [9] Fujitsu. Interconnect topology-aware resource assignment. <http://www.fujitsu.com/global/Images/technical-computing-suite-bp-sc12.pdf>.
- [10] Y. Georgiou and M. Hautreux. Evaluating scalability and efficiency of the resource and job management system on large HPC clusters. In *JSSPP 2012, Shanghai, China*, pages 134–156, 2012.
- [11] T. Hoefer and M. Snir. Generic Topology Mapping Strategies for Large-Scale Parallel Architectures. In *ICS*, pages 75–84, 2011.
- [12] E. Jeannot and G. Mercier. Near-optimal placement of mpi processes on hierarchical numa architectures. *Euro-Par 2010-Parallel Processing*, pages 199–210, 2010.
- [13] E. Jeannot, G. Mercier, and F. Tessier. Process Placement in Multicore Clusters: Algorithmic Issues and Practical Techniques. *IEEE Trans. Parallel Distrib. Syst.*, 25(4):993–1002, 2014.
- [14] G. Lee, N. Tolia, P. Ranganathan, and R. H. Katz. Topology-aware resource allocation for data-intensive workloads. In *APSys '10*, pages 1–6, 2010.
- [15] C. Liu, L. Yang, I. Foster, and D. Angulo. Design and evaluation of a resource selection framework for grid applications. In *HPDC '02*, pages 63–, 2002.
- [16] G. Mercier and J. Clet-Ortega. Towards an Efficient Process Placement Policy for MPI Applications in Multicore Environments. In *EuroPVM/MPI*, pages 104–115, Espoo, Finland, Sept. 2009.
- [17] G. Mercier and E. Jeannot. Improving MPI Applications Performance on Multicore Clusters with Rank Reordering. In *EuroMPI*, pages 39–49, Santorini, Greece, Sept. 2011.
- [18] J. Navaridas, J. Miguel-Alonso, F. J. Ridruejo, and W. Denzel. Reducing complexity in tree-like computer interconnection networks. *Parallel Computing*, 36(2-3):71–85, 2010.
- [19] Oracle. Grid engine.
- [20] PBSWorks. Pbs. <http://www.pbsworks.com/PBSProduct.aspx?n=PBS-Professional&c=Overview-and-Capabilities>.
- [21] R. Raman, M. Livny, and M. Solomon. Matchmaking: Distributed resource management for high throughput computing. In *HPDC'97*, Chicago, IL, July 1998.
- [22] M. J. Rashti, J. Green, P. Balaji, A. Afsahi, and W. Gropp. Multi-core and network aware MPI topology functions. In *EuroMPI 2011, Santorini, Greece*, pages 50–60, 2011.
- [23] C. A. Santos, A. Sahai, X. Zhu, D. Beyer, V. Machiraju, and S. Singhal. *DSOM 2004, Davis, CA, USA, November 15-17*, chapter Policy-Based Resource Assignment in Utility Computing Environments, pages 100–111. 2004.
- [24] C. Smith, B. McMillan, and I. Lumb. Topology aware scheduling in the lsf distributed resource manager. In *Proceedings of the Cray User Group Meeting*, 2001.
- [25] O. Sonmez, H. Mohamed, and D. Epema. Communication-aware job placement policies for the koala grid scheduler. In *e-Science'06*, pages 79–86, Dec 2006.
- [26] J. Wu, X. Xiong, and Z. Lan. Hierarchical task mapping for parallel applications on supercomputers. *The J. of Supercomputing*, 71(5):1776–1802, 2015.
- [27] X. Yang, Z. Zhou, W. Tang, X. Zheng, J. Wang, and Z. Lan. Balancing job performance with system performance via locality-aware scheduling on torus-connected systems. In *Cluster'2014*, pages 140–148, 2014.
- [28] A. Yoo, M. Jette, and M. Grondona. Slurm: Simple linux utility for resource management. In *Job Scheduling Strategies for Parallel Processing*, pages 44–60. 2003.
- [29] H. Yu, I.-H. Chung, and J. Moreira. Topology mapping for blue gene/l supercomputer. In *Supercomputing'06*, New York, NY, USA, 2006. ACM.

# Optimized Offloading Using Local Clusters

Hemant Tiwari  
 Samsung Research Institute  
 Bangalore, India  
 h.tiwari@samsung.com

Prem Kumar  
 Samsung Research Institute  
 Bangalore, India  
 prem.kumar@samsung.com

Balaji V. Ramalingam  
 Samsung Research Institute  
 Bangalore, India  
 balajiv@samsung.com

## ABSTRACT

This paper explores the process offloading in Mobile and Wearable devices to optimally use computational and power resources. It proposes the Performance and Energy Models over multiple devices to quantify the benefits and assist the decision to offload. It also presents a scheduler to maximize performance and tests with mixture of Data Exchange Intensive and Performance Intensive programs. The results can be extended to environments with partially predictable mobility.

## CCS Concepts

- Computing methodologies ~ Distributed Computing methodologies;

## Keywords

Process Offloading; Elastic Execution; Mobile and Wearable process offloading; Offloading Performance Model; Offloading Energy Model.

## 1. INTRODUCTION

The complexity, scope and computation needs of mobile and wearable applications are increasing every year. The expectation of users from these devices is also increasing. In this scenario, Process Offloading can be used to enhance the capability of mobile and wearable devices by migrating computations to other available and participating devices. With the availability of high bandwidth technologies, it makes sense to enable process offloading with growing number of devices in home with idle processors. The main contributions of this paper are as follows:

1. Performance and Energy Models for multiple devices to quantify the input for Offloading Decision Problem.
2. Enable Offloading using MPI for Devices.
3. Design and Validation of a scheduler based on Performance Model.

The paper is organized as follows:

We discuss the overview of mobile process offloading research efforts in Section 2. The architecture and analysis of offloading is presented in section 3.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [Permissions@acm.org](mailto:Permissions@acm.org).

*ICDCN '17, January 04-07, 2017, Hyderabad, India  
 © 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00*

DOI: <http://dx.doi.org/10.1145/3007748.3007785>

In Section 4 we present the results of offloading between Mobile to Mobile and Wearable to Mobile with different metrics. The implementation and results of a scheduler to achieve maximum performance is in Section 5. Section 6 discusses about our work in continuation and future scope.

## 2. OVERVIEW

Process Offloading has been existing in some form or other in computer industry, in form of Load Balancing, Grid/Cluster Computing etc. Term cyber foraging was used by Balan et al [1] for mobile devices which made transient and opportunistic use of more resourceful servers. Several frameworks such as Scavenger [2], MAUI [3], CloneCloud [4], Jade [5] and ThinkAir [6] have been suggested, but they do not utilize the standardized and tested MPI [7] methods and use their own specific language enhancements and compilers. These frameworks also generally use offloading to cloud. Offloading can be done to improve performance or to increase the battery life or having a balance between the two.

For Android devices, there are some big distributed computing projects e.g. BOINC (Berkeley Open Infrastructure for Network Computing) [8] and HTC Power to Give [9], these are distributed computing platforms used to harness the idle CPU cycles for big research projects e.g. SETI.

Offloading may not always beneficial, so there is a need to find out if offloading would result in better performance. In case there are multiple configurations available for offloading, the configuration resulting in best performance needs to be identified.

## 3. ARCHITECTURE AND ANALYSIS

In Figure 1. we present the High Level Architecture, which summarizes the necessary components of an offloading framework.

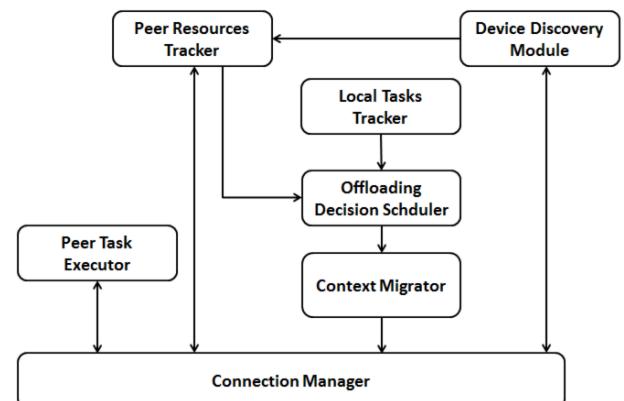


Figure 1 High Level Architecture

The Local Tasks tracker keeps track of tasks currently executing locally and the idle CPU. It broadcasts the idle CPU status with other resource parameters (e.g. Free RAM, Available Battery) using Connection Module. This broadcast can be done over BLE Advertisement Packet. The Device Discovery Module listens to BLE Advertisement Packets from other participating devices and extracts the information and sends to Peer Resource Tracker. Peer Resource Tracker keeps updated information of Peer Device Resources.

Using information of Local Tasks and Peer Resources, the Offloading Decision Scheduler takes the decision of offloading. The offloading is done using Context Migrator, Offloading Decision Scheduler will be explored in Section 5. Peer Task Executor takes care of receiving the Task and Context, computation and sending the result back. Connection Module abstracts connection related details of discovery, connection, reconnection etc. using any of the available connection technologies (e.g. BLE, Wi-Fi, WFD etc.).

From this offloading framework, we specifically focus on Offloading Decision Scheduler (ODS). This scheduler can decide to optimize performance or battery life or a balance in between. Although the assumption taken is that computations are fully parallelizable, the models can be extended to partially parallelizable computations. The Performance and Energy Models are needed by ODS are as follows:

### 3.1 Performance Model

Suppose  $S_{LOCAL}$  is the size of computation and  $C_{LOCAL}$  is the computation speed of local processor. The time to execute  $T_{LOCAL}$  in this case will be

$$T_{LOCAL} = S_{LOCAL}/C_{LOCAL} \quad \text{Eq. 1}$$

Suppose  $S_{OFFLOAD_i}$  is the size of computation offloaded to a peer device  $i$  with computation speed  $C_i$ . The time needed for this portion of task to be executed on peer device will be  $S_{OFFLOAD_i}/C_i$ .

The time needed to send the computation data back and forth device  $i$  also needs to be accounted, assuming the data transfer is delegated to separate network card/chip and does not consume main CPU resources. The time to transfer computation data depends on its size  $D_{OFFLOAD_i}$  and available bandwidth  $B_i$ . Hence total time  $T_{OFFLOAD_i}$  needed to send and get back result of computation from device  $i$  is:

$$T_{OFFLOAD_i} = \frac{S_{OFFLOAD_i}}{C_i} + \frac{D_{OFFLOAD_i}}{B_i} \quad \text{Eq. 2}$$

The time  $T_{OFFLOADlocal}$  taken by local processor after offloading is:

$$T_{OFFLOADlocal} = \frac{S_{LOCAL} - \sum S_{OFFLOAD_i}}{C_{LOCAL}} \quad \text{Eq. 3}$$

The Total Time  $T_{OFFLOADED}$  to complete the task is dependent upon the slowest step. The slowest step can be at local execution or slowest step from offloaded devices.

$$T_{OFFLOADED} = MAX [T_{OFFLOADlocal}, MAX (T_{OFFLOAD_i})] \quad \forall i \quad \text{Eq. 4}$$

The Offloading Performance Gain is

$$O_{PerformanceGain} = T_{LOCAL}/T_{OFFLOADED} \quad \text{Eq. 5}$$

It can be seen that offloading may be detrimental if one of the  $T_{OFFLOAD_i}$  causes  $T_{OFFLOADED}$  to increase. The aggregate nature of Eq. 4 establishes that the offloading has to be seen as an aggregated optimization problem.

If the devices in the environment are not expected to change frequently, then  $C_i$  and  $B_i$  can be known. Also in case it's the same tasks which need to be offloaded, then  $S_{LOCAL}$ ,  $S_{OFFLOAD_i}$  and  $D_{OFFLOAD_i}$  become predictable. With some

historical data, the identification of best performance scenario becomes possible.

### 3.2 Energy Model

Let the power needed for computation be  $P_C$ . Power needed to transmit/receive be  $P_{TxRx}$ . Idle power be  $P_{IDLE}$ .

If the task executes locally, the energy consumed  $E_{LOCAL}$  is

$$E_{LOCAL} = P_C * T_{LOCAL} \quad \text{Eq. 6}$$

In case of offloading, the energy consumed  $E_{OFFLOADED}$  is:

$$E_{OFFLOADED} = P_C * T_{OFFLOADlocal} + P_{TxRx} * \sum \frac{D_{OFFLOAD_i}}{B_i} + P_{IDLE} * T_{IDLE} \quad \text{Eq. 7}$$

$$T_{IDLE} = MAX(MAX(T_{OFFLOAD_i}) - T_{OFFLOADlocal}, 0) \quad \text{Eq. 8}$$

The Offloading Energy Gain  $O_{EnergyGain}$  is:

$$O_{EnergyGain} = E_{LOCAL}/E_{OFFLOADED} \quad \text{Eq. 9}$$

Since the parameters  $P_C$ ,  $P_{TxRx}$  and  $P_{IDLE}$  can be known as they are local, the  $O_{EnergyGain}$  can be known once  $O_{PerformanceGain}$  is known.

## 4. TESTS AND RESULTS

To Test the offloading, the target devices chosen were Android Phones and Tizen Wearable devices. To write parallel programs MPICH2[10] based on MPI [11] standard was used. The MPICH2 is available for Linux systems and uses *glibc* [12] library, while Android uses *bionic*, a lightweight library. For usage on Android and Tizen, MPICH2 needs to be cross compiled for ARM and statically linked with *uClibc* [13] which is the C library for embedded systems. This was done by creating ARM toolchain using *BuildRoot* [14].

To test the implementation, various types of programs were tested with different computation and data exchange characteristics. e.g. Distributed Prime Calculation: Parallelizable, Computation Intensive, Very Limited Data Exchange needed.

Distributed Merge Sort: Parallelizable, Computation Intensive, Significant Data Exchange needed. [15]

The setup uses static offloading i.e. decision to offload is taken prior to running the computation. Offloading medium is Wi-Fi and computations are migrated to nearby devices without cloud.

### 4.1 Distributed Prime Calculation

Figure 2 depicts the performance of offloading prime calculation in single vs. 2 to 5 devices.

X axis: Number till which all primes have to be calculated.

Y axis: Time Taken in Seconds.

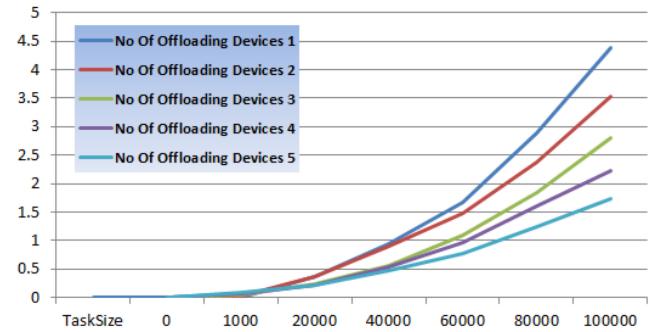
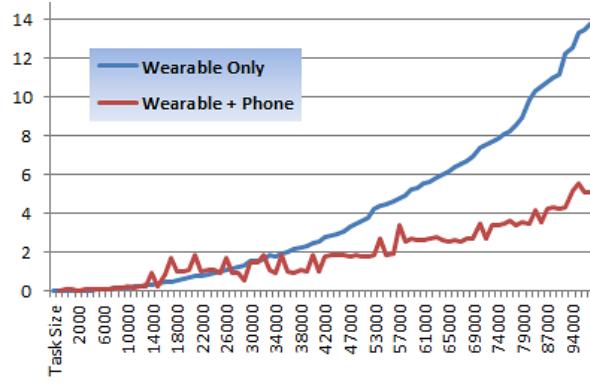


Figure 2 Offloading Prime Calculation in Android Devices

It is interesting to note the positions of intersection of curves. Since the data exchange is limited, the time needed is directly proportional to size of tasks. At smaller task sizes, the data exchange time exerts an influence and offloading to more device may be detrimental, as the size of task increases, advantage of offloading with more devices increases.

Fig.3 depicts the performance of offloading prime calculation in wearable alone and when both wearable and phone is used.

X axis: Number till which all primes have to be calculated.  
Y axis: Time Taken in Seconds.



**Figure 3 Offloading between Wearable and Phone**

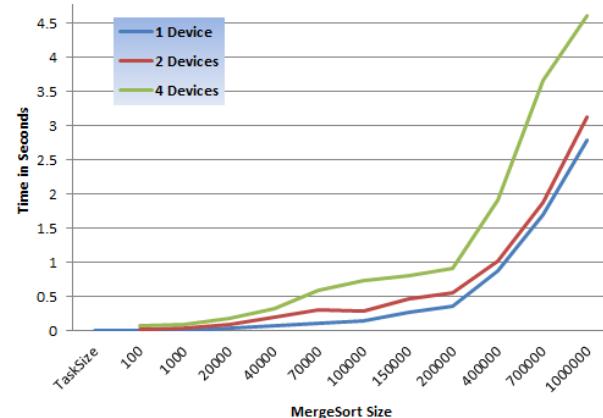
As observed in above figure, there is earlier distinct advantage of offloading computationally intensive task from a CPU constrained device to a device with high CPU capacity. The Blue line is smooth while the red line is jagged, this may be attributed Jitter in Wi-Fi network.

## 4.2 Distributed Merge Sort

Distributed Merge Sort has much higher data exchange requirement. Figure 4 depicts results of performing merge sort locally vs offloading with 2 and 4 devices.

X axis: Number of elements in list to be sorted.

Y axis: Time Taken in Seconds.



**Figure 4 Distributed Merge Sort**

As observed in Figure 4, Performance degradation is observed when the number of devices to offload are increased. This degradation becomes severe if size of list is increased, e.g. for sorting 1 million elements, local execution takes 2.78 seconds, sorting using offloading between 2 and 4 devices take 3.1s and 4.6

seconds respectively. This may be attributed to the fact in merge sort needs much higher transfer of data, E.g. for offloading to a single device, the computational gain is just  $\frac{n}{2} \log \frac{n}{2}$  but the data transfer totals n (Send n/2 points to peer and receive n/2 sorted points) resulting in  $\frac{D_{OFFLOADi}}{B_i}$  becoming a dominating factor. It is also evident that a solution to problem may be using technologies with higher  $B_i$  (Bandwidth).

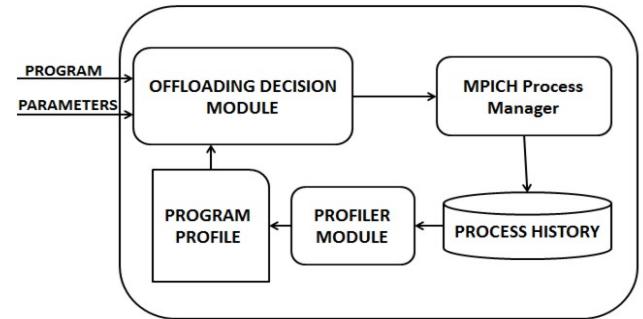
## 5. PERFORMANCE SCHEDULER

It is evident form the test results in section 4 that the decision to offload and to how many devices mainly depends on following:

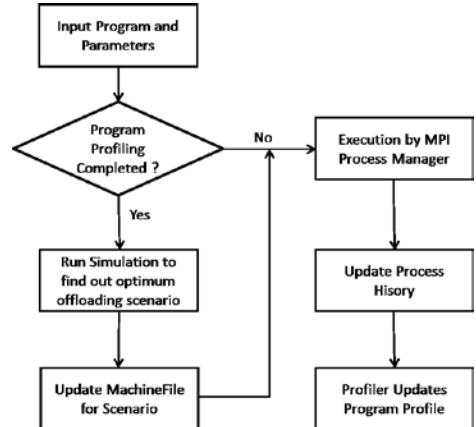
1. Computation Size
2. Data Exchange.
3. Processor Speeds.
4. Bandwidth.

Since every program has different Computation Size and Data Exchange Needed, to achieve maximum performance, the programs need to be profiled. Based on discussed performance model discussed in Section 3.1, we implemented and tested an Offloading Decision Scheduler to achieve maximum performance.

As depicted in Figure 5, the program to be executed and parameters if any are passed to scheduler. The Scheduler first builds a profile of a program by fitting the actual execution history over Performance Model to derive  $S_{LOCAL}$  w.r.t. input parameters,  $D_{OFFLOADi}$ ,  $C_{LOCAL}$ ,  $C_i$ , and  $B_i$ . Once these values are derived by solving equations, the scheduler takes decision of offloading to achieve maximum  $O_{PerformanceGain}$  using simulation (Figure 6).



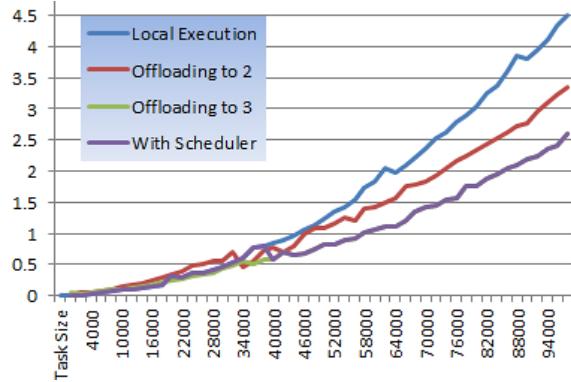
**Figure 5 Scheduler Modules**



**Figure 6 Scheduler Decision Chart**

The test program chosen for scheduler was Prime Calculation. Figure 7 shows the results with using scheduler to offload :

X axis: Number till which all primes have to be calculated.  
Y axis: Time Taken in Seconds.



**Figure 7 Optimized Offloading Using Scheduler**

The Scheduler was first trained with sample runs to build a program profile. The program profile as of now consists of program and device parameters in Performance Model. Once profile is ready, for each new run, simulation is done to predetermine configuration with best performance. The machine file (a MPI configuration file to determine offloading) is updated to configure MPI process manager with optimum scenario. As seen, scheduler decides to locally execute the task till input size is  $\sim 40000$  (overlaid blue curve), and then decides to offload to 3 devices (overlaid green curve). Thus, in case of multiple configurations available for offloading, the scheduler can identify configuration resulting in best performance.

## 6. CONCLUSION AND FUTURE WORK

In this paper, we investigated offloading w.r.t. Performance and Energy Models and presented simple aggregated equations for these Models. The Architecture and method to enable the offloading via MPICH2 was discussed. We tested and explained our Performance Model on basis of test results on Tizen Wearables and Android Devices.

Investigation of various aspects of offloading by testing with Data Exchange Intensive and Computation Intensive programs was done. Implementation and testing of the Scheduler Based on Performance Model was done.

We will be continuing our work to verify the Energy Model and testing by designing a scheduler using Energy Model.

We hope that our findings will be useful to enable further offloading scenarios in Mobile, Wearable and other Low Power Devices. We are expanding our research to create an ecosystem in which offloading helps user to utilize the full potential of available devices (Wearables, Smart TV, Extra Phones) by forming an Elastic Computation Cluster locally, with optimal offloading intelligence.

## 7. ACKNOWLEDGEMENTS

The authors wish to thank Raju Dixit, Sreevatsa D.B., Sukumar Mohrana, Vanraj Vala, Amit A. Mankikar and Sibsambhu Kar from Samsung Research Institute for their valuable inputs and suggestions for this paper

## 8. REFERENCES

- [1] R. Balan, J. Flinn, M. Satyanarayanan, S. Sinnamohideen, and H.-I. Yang. The Case for Cyber Foraging. In The 10th ACM SIGOPS European Workshop, Saint-Emilion, France, September 2002.
- [2] M. Kristensen, "Scavenger: Transparent development of efficient cyberforaging applications," Proceedings of IEEE International Conference on Pervasive Computing and Communications, March 2010.
- [3] Cuervo E, Balasubramanian A, Cho D, Wolman A, Saroiu S, Chandra R, Bahl, MAUI: making smartphones last longer with code offload. International conference on mobile systems, applications, and services. 2010
- [4] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. Clonecloud: Elastic execution between mobile device and cloud. EuroSys Proceedings of the sixth conference on Computer systems, 2011. DOI 10.1145/1966445.1966473
- [5] Hao Qian, Daniel Andresen, Jade: An Efficient Energy-aware Computation Offloading System. 15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing(SNPD),2014.DOI: 10.1109/SNPD.2014.6888703.
- [6] Kosta, S., Aucinas, A., Hui, P., Mortier, R., & Zhang, X. (2012, March). Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In INFOCOM, 2012 Proceedings IEEE (pp. 945-953). IEEE.
- [7] <https://computing.llnl.gov/tutorials/mpi/>
- [8] <http://boinc.berkeley.edu/>
- [9] <https://play.google.com/store/apps/details?id=com.htc.ptg>
- [10] <http://www.mcs.anl.gov/project/mpich-high-performance-portable-implementation-mpi>
- [11] MPI: A Message-Passing Interface Standard Version 3.0, Message Passing Interface Forum, Sep 2012
- [12] <https://www.gnu.org/software/libc/>
- [13] <https://www.uclibc.org/>
- [14] <https://buildroot.org/>
- [15] Atanas Radensk, Shared Memory, Message Passing, and Hybrid Merge Sorts for Standalone and Clustered SMPs. The 2011 International Conference on Parallel and Distributed Processing Techniques and Applications PDPTA'11

# Improved Write Access Control and Stronger Freshness Guarantee to Outsourced Data

Naveen Kumar  
IIIT, Vadodara  
Gujarat, India

naveen\_kumar@iitvadodara.ac.in

Anish Mathuria  
DAIICT, Gandhinagar  
Gujarat, India

anish\_mathuria@daiict.ac.in

## ABSTRACT

In secure data outsourcing, even a single unauthorized write operation may heavily ruin the data owner's business operations. Another potential issue is that the untrusted service provider may return stale data to mislead the readers. Existing work on secure write access does not consider misbehavior by users with write authorization to the outsourced data files. The data owner may not wish to allow such users to modify their own written data files after a fixed amount of time. The other type of misbehavior is that a user with revoked access to a resource can modify corresponding latest versions of the resource written by him in collusion with the service provider. Also, the weak freshness guarantee (i.e., the  $< k, t >$ -staleness) is not appropriate in many time-sensitive applications.

In this work, we address the misbehavior by users who attempt to modify the own written data files. Such misbehavior is detected by the data owner using an audit-based mechanism. The viability of the proposed mechanism is verified by implementing the proposed protocols in Microsoft's Azure platform. A strong freshness guarantee is addressed using proof messages from the service provider so that a read request will return the latest version of the data file at least until the time when the data file is sent from the service provider to a reader.

## CCS Concepts

•Information systems → Distributed storage; •Security and privacy → Access control; Authorization;

## Keywords

Write access, data outsourcing, write-serializability, freshness

## 1. INTRODUCTION

In this paper, we consider an outsourcing scenario where a data owner encrypts its files and uploads them to an untrusted cloud service provider (CSP). We focus on symmetric encryption to protect data that is stored at CSP. Each data file is encrypted with a distinct secret key. The keys needed to decrypt the files

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ICDCN '17, January 04-07, 2017, Hyderabad, India

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007778>

are distributed to authorized users who can download the outsourced encrypted files directly from the cloud server, decrypt and use them. For proper maintenance and security reasons, an outsourced record contains an encrypted data file along with its metadata. Write access control means that only write authorized users can perform the write operation. The CSP handles write access requests directly from the users without interacting with the data owner (as in read access).

In a distributed setting, one user might be reading a record at the same time that the other is trying to update the same record. This may lead to an inconsistent view of data. To avoid this problem "versioning" [11] is used. Versioning is about assigning sequence numbers to the committed updates for a resource. Here, an update means writing a new version of a data file. Readers can see the old committed versions of a data file and once the current update is committed, only then the reader can see a new version of the data. The serializability property ensures that each new update is written over the latest committed version, even when the service provider receives multiple updates over the same committed version.

As the CSP is untrusted, it may modify the sequence of updates maliciously. This modified sequence of updates may mislead the readers. A well-known attack is "fork attack" where the untrusted service provider attempts to hide a legitimate user's update [14]. A property named fork consistency defends against fork attacks [14]. It guarantees that at a given point of time all the users will view the same resource's version. A property stronger than fork consistency is "write-serializability" [16]. It ensures a stronger link between every contiguous pair of resource versions, allowing the data owner to detect tampering of the sequence of updates.

In a distributed cloud scenario replicas of data are created and stored at cloud servers situated at different geographical locations. Ideally, any data update must be reflected over every corresponding replica immediately. In practice, there are delays in updating the replicas. This may lead to stale data view (without any misbehavior) to the readers who fetch the data from replicas that have not yet received the updates.

## 1.1 Motivation

In what follows, we motivate two specific requirements on write access control and freshness.

### *Write access control.*

Suppose that an authorized user Alice has just performed a write operation on some file. Further, suppose that no other user has performed a write operation on that file after Alice's write. In this situation, the CSP may misbehave by allowing Alice to modify the content of the file without creating a new version. Mod-

ifying a file here means altering the content of the file. This is a matter of concern even if the user is an authorized writer. For example in a cloud-based e-newspaper publishing system, the news is created by staff writers and uploaded to a third party storage service provider. These news articles are then published. The e-newspaper CEO will not allow any modification to the content after it is published. However, it may happen that the published news article has some unauthenticated content which may lead to some legal action or embarrass the newspaper. Meanwhile, let the staff writer collude with the service provider and modify the published content at the server to avoid any legal consequences. Now nobody including the CEO can frame any charges against the staff writer as outsourced news is modified.

Similarly, in the case of user's access right revocation i.e., the revoked user can collude with the CSP and modify their latest committed version (s) of a resource whose access right is now revoked. Such unauthorized write becomes possible because the record is not updated by the data owner after its access right is revoked and old write authorized keys of the record are available with its writer.

#### *Data freshness.*

In case the staleness is maliciously injected by the CSP, it must be caught by the data owner. For example, in a stock market application, a small delay in updating the bid price can cause a huge loss to a bidder. Similarly, a small delay in getting the current status of seat allocation in a railway reservation system may prevent a user from getting a confirmed seat reservation. Therefore, it is desirable to reduce the read staleness (or improving freshness guarantee).

## 1.2 Contributions

*Auditing* is a well-known misbehavior detection mechanism. It will detect the misbehavior at some later time and take an appropriate action to avoid it in future. In data outsourcing scenario, it is executed by the data owner at regular intervals of time.

In this work, we propose an audit-based mechanism to address the above requirements so that a user even after collusion with the CSP is not able to modify self-written outsourced data files, after a given amount of time or after access right revocation. If such modification happens, it will be caught during the audit process by the data owner and charges can be framed against the misbehaving party. Existing works ([13, 17, 16, 8]) do not consider this requirement.

We examine the staleness problem and improve on the freshness guarantee as compared to the existing works. Existing works ([10, 8]) guarantee that the resource version was fresh till the time when it was written by a user. Our proposed mechanism ensures that file retrieved from the cloud server is fresh at least until the time when the file is dispatched by the CSP to the reader. In case the stale data is dispatched by the CSP during a read request, it will be caught by the data owner. The proposed mechanisms are implemented on Microsoft Azure platform and the results show that the suggested mechanisms are viable in practice.

## 1.3 Organization

In the following section, we describe our system model and related security requirements. Section 3 will discuss the related work followed by the existing mechanisms used for implementing the given security requirements. In Section 4, we review the existing schemes with respect to the security of write access against users who collude with the CSP to modify the content of their files. We highlight their shortcomings and give appropriate coun-

termeasures. Implementation of the modified system is given in Section 4.2. Section 5 address the proposed improved freshness notion in detail. Section 6 concludes this work.

## 2. SYSTEM MODEL

As shown in Figure 1, we consider a system with three entities: the data owner, CSP and the users. The user or data management information will be sent by the data owner to the CSP and write access authorization is sent to the authorized users. The CSP is responsible for handling write access requests directly from the authorized users. The users (for example, the employees of an organization) are assumed to be trusted by the data owner. However, users may involve in unauthorized activities to protect themselves. For example, if any unauthorized content is written by a user, he may try to restore the record to a previous version with the connivance of CSP. On the other hand, we consider malicious but cautious CSP which launches no attack that leads to any provable trace [2]. It will include active attacks.

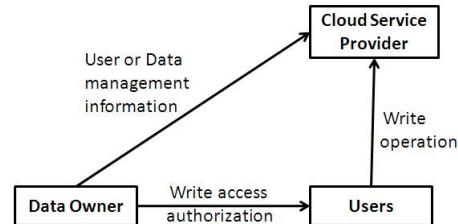


Figure 1: Cloud architecture for write access

In our setting, a user can have both read and write access permissions to the outsourced files. We assume that a user can have write access only if she has read access i.e., a user can read and modify the existing file content. A user will access the outsourced resource directly from CSP without interacting with the data owner. As the CSP is untrusted, it may allow write access to an unauthorized user. If such user writes or modifies data, it may ruin the data owner's public image or affect the normal functioning of the organization. To mitigate the problem, some proof is needed to convince the data owner that no unauthorized write operations have been allowed by the CSP ([16]). It must be noted that if CSP is honest then such proof is not required.

The data is cloned over different servers to allow fast data access from a closest server and recover data in emergency situations such as server crash. Data can be accessed by a number of authorized users concurrently. At the time of write access, a data file is updated by a write authorized user at one server and then replicas at various servers are updated by the CSP. Therefore, a sequence of updated versions is stored for each data file. For security reasons, we allow only the data owner to delete the files. In our threat model, a user may attempt to access the data for which he is not authorized. Possibly he may collude with the CSP for unauthorized data access. Moreover, the data owner or users may accuse the CSP of violating system security properties.

A file update operation works as follows: the file is first read by an authorized user who will then update the file locally and send the updated version back to the server for storage. A user can initiate two types of transactions: *get* and *put*. We define a transaction to mean a sequence of operations each treated as a unit such as read, write and commit. In *get* transaction, the user will first send read request to the CSP which verifies the request and responds back with the requested resource, if the request is valid.

Otherwise, the CSP will respond with “invalid request” message. In *put* transaction, the user will first create a new updated version and send it to the CSP. Upon receiving the new version, the CSP will authenticate the user and verify the updated version. If verified correctly, the new version is stored at the server and an acknowledgment is returned to the writer. In readers view, a *get* transaction is called committed when it receives the requested data file. In writers’ view, a *put* transaction is called committed only when it receives the acknowledgment. If the write request is disallowed, the CSP will return a negative acknowledgment.

## 2.1 Security requirements

We consider the following requirements for secure data outsourcing [16, 4, 8].

### *Data confidentiality.*

This property ensures that no unauthorized user including the CSP can read the outsourced information. This requirement is important when an organization outsources its secret data to an untrusted CSP.

### *Data integrity.*

This property ensures that no unauthorized user (including the CSP) can update or modify a resource even if it colludes with other unauthorized users.

### *Write-serializability.*

This property ensures that all users will see all versions of a resource in the same order as they are written by various users [16]. In other words, this property ensures that a resource version  $i$  is always written over version  $i - 1$ . It implements a strong binding between each contiguous pair of data versions so that the CSP is unable to hide the sequence of updated versions for a resource.

### *Freshness.*

This property ensures that a read operation will always return the latest committed version of the requested resource. This requirement needs to be relaxed in a distributed environment since it cannot be fully satisfied (as discussed above). However, it is desirable to minimize the staleness.

### *Secure revocation.*

A user whose access rights are revoked should not be able to access any new or modified file which is published or modified after revocation. For example, consider a user whose access right is revoked for resource  $r$  at time  $t$ . The user is no longer authorized to access  $r$  for any time  $t_1$  with  $t_1 \geq t$ .

## 3. BACKGROUND

### 3.1 Related Work

Early cryptographic file systems with untrusted servers proposed by Cattaneo et al. [6] and Miller et al. [15] provides data integrity by storing a hash value and digital signatures for each data file. There are schemes which provide high-level integrity for cloud-based file system i.e., Proofs of Retrievability (PoR) [12, ?]. Stefanov et al. [?] proposed dynamic PoR enabled through an auditing protocol that continuously monitors the correctness and availability of the entire file system. In this research work, we are however considering low-level integrity and strong freshness semantics for a malicious storage system.

Li et al. [13] scheme proposes a network file system that ad-

dresses data integrity and consistency without trusting the storage servers. A weaker consistency model named “fork consistency” is implemented using a central Version Structure List (VSL). It is stored and updated by the server on every write update request. The VSL only contains the latest version structures i.e., there is no strong linkage between the sequence of versions. Shraer et al. [17] scheme guarantees integrity and consistency using an untrusted verifier implement the consistency and integrity check functions. The scheme requires an additional message exchange between the user and the verifier for each read and writes operation. Recent schemes (for example, [16, 8]) implement stronger property named write-serializability using “chain hash”. They provide strong linkage between all the data versions so that no unauthorized user including the CSP can modify the sequence. Popa et al. [16] consider proof messages generated during each read/write access and are used by the data owner at the time of audit (as in [?]) to prevent any inconsistent behavior by the CSP.

For freshness, Popa et al. [16] scheme uses a random nonce  $N$  generated and sent by the user with read request which the CSP returns in the response message. The presence of  $N$  in the message assures that the response is computed afresh. The proof messages commit that the current version ( $i$ ) is fresh as compared to the old versions ( $i - 1, i - 2, \dots$ ). It cannot guarantee that a new version is written between the time when  $i$ th version was written and when it was read by the user. In SiRiUS [10], for each file the meta-data (contains access control information) is continuously signed by the user with updated timestamp (for freshness). The timestamp ensures the time when data record was written. Similarly, in [8], freshness is provided by timestamp stored with each data record and encrypted by shared secret key is known only to write authorized users so that no unauthorized user can modify it.

The schemes [16, 8] address the access right revocation problem without using any separate verifier as in [17]. Popa et al. [16] uses lazy revocation ([9]) to handle access right revocation i.e., a revoked file is re-encrypted only when the file is modified for the first time after being revoked. Therefore, the CSP can collude with the writer to modify the last written file until the file is updated. In [8], write access revocation is handled by updating the secure write access token at CSP. However, it will not restrict the untrusted CSP (in collusion with the writer) from modifying the existing resource.

Table 1 summarizes the mechanisms used by existing schemes with respect to integrity (I), write-serializability (W) freshness (F) and secure revocation (R)). In the table, integrity is addressed using Message Authentication Code (MAC [5]) in the latest two schemes ([16, 8]). Chain hash is preferred over the global lock and out-of-band communication-based mechanisms for handling write-serializability by the recent schemes. The scheme in [16] provides version-based staleness i.e., a read returns one of the  $k$ th latest version.  $k$  is fixed in a system and depends on the time required by a data record to reach a reader. In the table, ‘X’ indicates that the security property is not addressed.

For freshness, Popa et al. [16] only assures the user about the freshness of the received message. Vimercati et al. [8] use timestamp (as a commitment message) stored securely with each record version that assures the time when it was written. However, in both the schemes, the CSP can still give the stale resource in response to a read request without the misbehavior being detected by the data owner.

### 3.2 Basic mechanisms

Here we review some of the existing mechanisms used in exist-

**Table 1: Comparison**

$\downarrow$ Scheme \ security properties $\rightarrow$	I	W	F	R
Li et al. [13]	Hash tree	Global lock	X	X
Shraer et al. [17]	Verifier	Out-of-band communication	X	X
Popa et al. [16]	MAC	Chain hash	Version-based staleness	Lazy revocation
Vimercati et al. [8]	MAC	Chain hash	X	Renew token

ing write access control schemes.

### 3.2.1 Audit mechanism based on MAC

A possible solution to address data integrity property is by using MAC and “audit” mechanism executed by the data owner to verify any unauthorized outsourced data modification [16]. A MAC computed with a secret key known only to the authorized users is stored with each outsourced record. Since the CSP is untrusted, it requires that no user except the data owner is allowed to delete any outsourced data record. If data deletion is allowed by such users then to prove any unauthorized data modification may become difficult to verify, without the knowledge of old (deleted) records. Also, proof messages can be given by the CSP against each read/write operation to assure the correctness of the operation. These messages are then collected at the data owner which will use them in the auditing process. It must be noted that if CSP is honest then the auditing mechanism is not required. Popa et al. [16] suggest time-bound auditing of outsourced data by the data owner. Priorities are assigned to the data files so that the data with higher priority is audited after each time interval whereas audit of lower priority data can be delayed.

### 3.2.2 Chain hash

Write-serializability property can be obtained by storing a secure message such as a “chain hash” with each record [16, 8], described as follows. Chain hash creates a strong binding between every pair of contiguous record versions so that no unauthorized user (including the CSP) can insert a record in-between. For example, consider a resource with versions  $v_1, \dots, v_i$  with the corresponding chain hashes  $CH_1, \dots, CH_i$  where  $CH_j = MAC_K(j^{th} \text{ resource information} + CH_{j-1})$  and  $K$  is symmetric key known to the writer who has written the updated record.  $K$  is also known to the data owner (who generates it) who can verify the chain hash during the auditing process. Since  $K$  is only known to the writer and the data owner, no other user including the CSP can modify the outsourced resource and create the corresponding modified chain hash using  $K$ .

### 3.2.3 Freshness proof

Ideally, the freshness ensures that a user will always get the latest modified data. The basic mechanism for detecting freshness violations is that for a read request, the CSP will return the requested record along with the commitment message as a freshness proof [16]. This message will be used later at the time of auditing by the data owner for verifying freshness of the message. If CSP had misbehaved by sending a stale record, the commitment message serves as a proof of its misbehavior.

## 4. IMPROVED AUDITING PROTOCOLS

Existing write access control schemes ([13, 17, 16, 8]) ensure that a file can be written only by write authorized users. However, they do not consider the following types of misbehavior.

1. A write authorized user in collusion with the CSP can mod-

ify his own written latest sequence of data records until a new version is written by another user.

2. A user whose access right is revoked, in collusion with the CSP, can still modify his latest committed version (s) of revoked resource (s).

In the above, the latest record of a resource refers to the last written version of the resource. Similarly, the latest records of a resource refer to the recent chain of versions (including the last version) of the resource.

For example, consider a resource  $r$  whose last version  $v_i$  was written by a user  $u$  at time  $t$ . Now, until a new version  $v_{i+1}$  is written, the user  $u$  can modify it by colluding with the CSP. This is possible even if the access permission for resource  $r$  is revoked from  $u$ . Also, if  $u$  has written the latest contiguous sequence of versions  $v_j, v_{j+1}, \dots, v_i$ , it can modify any number of versions in that sequence. In the existing schemes, there is no mechanism to enable the data owner to obtain proof of such misbehavior.

In the following, we propose two protocols to mitigate the above threats. We assume that the system time is divided into equal sized distinct time slots (as in [16]). A secret symmetric key  $K_s$  is assumed to be known only to the data owner.

In *Protocol 1*, the data owner will verify and attest each of the latest committed records of various resources written in the current time slot with its secret key  $K_s$ . The attestation is such that if the integrity of any old committed record of the attested record is breached, it must be caught by the data owner during the audit process. In *Protocol 2*, immediately before a user’s access right is revoked, the data owner will attest each of his latest committed records written in the current time slot. It ensures that the integrity of revoked resource will be intact even if the CSP colludes with the writer of the resource. Encryption keys can be updated as in lazy-revocation ([9]) method.

**Protocol 1:** The protocol for handling the first threat is described in algorithm 1. It runs between CSP and the data owner (DO). The goal is to ensure that even the authorized user cannot modify their own written data records after a pre-defined amount of time i.e., after the end of the current time slot. To achieve this, at the end of each time slot the CSP will call Algorithm 1.

In Algorithm 1,  $t_i$  denotes the current time slot. At the end of each  $t_i$ , the CSP will create a set  $S$  of triplets corresponding to each latest data record version written in the current time slot (Step 2). A triplet corresponding to resource  $o$  contains resource identifier  $ID_o$ , its latest version number  $ver_j$  and corresponding chain hash ( $CH_j$ ). Then the CSP will compute the hash of set  $S$  and  $t_i$  (to avoid replay attack). This hash is then signed by the CSP using its private key  $Pr_{CSP}$ . The signed hash is used by the data owner as a correctness proof for the current update operation. The signed hash and the set  $S$  are then sent to the data owner (Step 3). Upon receiving the message, the data owner will first compute and verify the signature (Step 4). If the signature is invalid, stop the process and request the CSP to send it again. If signature is correct then for each  $CH_j$  in  $S$ , the DO will do

---

**Algorithm 1** *ControlledWriteAccess()*

---

```

1:  $t_i \leftarrow$  current time slot
   /* Create set  $S$  containing triplets <resource id, version number, chain hash (CH)> */
2: CSP creates set  $S \leftarrow \{< ID_o, ver_j, CH_j >\}$  for each latest record committed in  $t_i$ 
3: CSP  $\rightarrow$  DO: Send  $S + \{h(S, t_i)\}_{Pr_{CSP}}$ 
4: DO verifies  $\{h(S, t_i)\}_{Pr_{CSP}}$ . If verification fails, stop the process and ask the CSP to send it again
5: for (each chain hash  $CH_j$  in  $S$ ) do
6:    $C \leftarrow \text{RetrieveChain}(ID_o, ver_j)$ 
7:   if ( $\text{CorrectChain}(C)$ ) then
8:     DO computes  $MAC_{K_s}(CH_j)$ , replace it with  $CH_j$  in  $S$ 
9:   else
10:    "Verification fails", DO takes appropriate action
11:   end if
12: end for
13: DO  $\rightarrow$  CSP: Send signed updated  $S$ 

```

---

**Algorithm 2** *RetrieveChain( $ID_o, ver_i$ )*

---

```

1: DO  $\rightarrow$  CSP: send get request with  $(ID_o, ver_i)$ 
   /* create sequence  $C$  containing latest records  $rec_{o,j}$  of resource  $o$  and version  $j$  */
2: CSP creates and initialize set  $S \leftarrow \{rec_{o,i}\}$ 
3:  $k \leftarrow i$ 
4: while ( $rec_{o,k-1}$  and  $\text{not } MAC_{K_s}(CH_{k-1})$ ) do
5:    $C \leftarrow C \cup \{rec_{o,k-1}\}$ 
6:    $k \leftarrow k - 1$ 
7: end while
8: if ( $rec_{o,k-1}$  and  $MAC_{K_s}(CH_{k-1})$ ) then
9:    $C \leftarrow C \cup \{rec_{o,k-1}\}$ 
10: end if
11: CSP  $\rightarrow$  DO:  $C$ 

```

---

the following: retrieve corresponding latest sequence of records by calling  $\text{RetrieveChain}(ID_o, ver_j)$  (Algorithm 2) and if it is correct then computes MAC over  $CH_j$  using key  $K_s$ . The newly computed MAC is now replaced with corresponding  $CH_j$  in  $S$  (Steps 6 – 8). In the function call,  $ver_j$  is the version number of resource  $o$  with id  $ID_o$  corresponding to the  $CH_j$ . Updated  $S$  is now sent to the CSP (Step 13). Upon receiving the updated  $S$ , the CSP will update the corresponding records at the server by replacing their *chainhashes* with received updated MACs (or *chainhashes*). It is to be noted here that if the *DO* is temporarily not reachable then the corresponding messages are stored in its mailbox. The protocol will be resumed whenever *DO* is online.

The  $\text{RetrieveChain}()$  method allows the data owner to retrieve the chain (i.e.,  $C$ ) of given resource' id ( $ID_o$ ) and version number ( $ver_j$ ) from the CSP. The chain  $C = \{rec_{o,k}, \dots, rec_{o,i}\}$  is such that the  $CH_k$  associated to record  $rec_{o,k}$  with version  $ver_k$  has MAC'ed with key  $K_s$ . Also, there is no  $CH_l$  exists with  $k+1 \leq l \leq i$  and MAC is computed with  $K_s$ .

Using  $\text{CorrectChain}()$  method (Algorithm 3), the data owner verifies the given chain hash of a given resource. If correctly verified then the algorithm returns "success" else it returns "failure".

In case the CSP does not send the latest version of a resource to the data owner, it will be caught by the data owner. There are two possible cases: only CSP misbehaves or the CSP will collude with the writer. In the first case since the writer is trusted, the timestamp will ensure the time of resource update. If the resource was written in the previous time slot, the CSP will be caught by

---

**Algorithm 3** *CorrectChain( $C$ )*

---

```

1: for (each record  $rec_{o,l}$  in  $C$ ) do
2:   DO verifies the  $CH_l$ 
3: end for
4: Return "True" if all chain hashes are correct, else return "False"

```

---

looking at the timestamp. If the resource is written in current time slot but is not the latest version then also it will be caught. It is because at the end of each time slot the latest version must be attested and an intermediate version is only attested if corresponding access right is revoked. It means if the latest version of a resource is not attested at the end of the current time slot, it cannot be attested later and will be caught easily by the data owner. If the writer colludes with the CSP and will modify the timestamp then also it will be caught by the data owner. It is because the timestamp is captured in the user tag of next version written for that resource. If there is no new version written then it will be treated as a new version.

**Protocol 2:** To handle the second threat, we propose a protocol described in Algorithm 4. The goal of this protocol is that a user whose access right is revoked for a resource (has written previously some latest records for the resource) cannot modify their own written records, even if it colludes with the CSP. Let a user  $u$  be revoked from accessing resource  $r$  (with identifier  $ID_r$ ). The data owner will call procedure  $\text{RevokeAccess}(u, ID_r)$  (i.e., Algorithm 4).

---

**Algorithm 4** *RevokeAccess( $u, ID_r$ )*

---

```

1:  $t_i \leftarrow$  current time slot
2: DO  $\rightarrow$  CSP: Send  $ID_r$ 
3: if (latest  $ver_j$  of resource  $r$  is written in  $t_i$ ) then
4:   CSP  $\rightarrow$  DO: Send latest record  $rec_{r,j}$  for  $ID_r$ 
5:   if ( $rec_{r,j}$  is written by  $u$  and  $\text{CorrectChain}(\text{RetrieveChain}(ID_r, ver_j))$ ) then
6:     DO computes  $X \leftarrow MAC_{K_s}(CH_j)$ 
7:     DO  $\rightarrow$  CSP: Send  $X$ 
8:     CSP will replace  $CH_j$  with  $X$ 
9:     CSP  $\rightarrow$  DO: Send "Acknowledgment"
10:  end if
11: else
12:   CSP  $\rightarrow$  DO: "no such record found"
13: end if

```

---

In  $\text{RevokeAccess}()$  method (Algorithm 4), the data owner will first send the read request for resource  $r$  to the CSP (Step 2). If latest version of the resource  $r$  is written in current time slot  $t_i$ , the CSP returns the corresponding record (Steps 3 – 4) else return "no such record found" (Step 12). If the received record was written by the revoked user, the data owner will compute MAC over it using secret key  $K_s$ , and send it back to the CSP (Steps 5 – 7). The CSP will update the received chain hash into corresponding resource record and return an acknowledgment back to the data owner (Steps 8 – 9).

The above-discussed shortcomings are formalized together as follows. In the definition below, the unauthorized user refers to untrusted cloud service provider, authorized users who have written some historical data and the revoked users.

**DEFINITION 1 (COLLUSION-SECURE).** A system is called collusion-secure if even in the presence of two or more unauthorized users (collude together), the secret keys known to all the users don't contain

---

**Algorithm 5** *Audit()*

---

```
1: DO → CSP: Prepare and send set  $S = \{< ID_o, ver_i >\}$  to  
be audited.  
2: for (each  $< ID, ver >$  in  $S$ ) do  
3:    $C \leftarrow \text{RetrieveChain}(ID, ver)$   
4:   Verify corresponding stored proof messages against  $C$ . If  
verification fails, appropriate action will be taken by DO.  
5:   if (Proof messages are correct) then  
6:     Return "Success"  
7:   else  
8:     Return "Failure  $< ID, ver >$ "  
9:   end if  
10: end for
```

---

any key that can be used in generating secrets that will allow any unauthorized write access in the system.

CLAIM 1. *Protocol 1 and Protocol 2 are collusion secure.*

PROOF. Upon receiving the updated sequence of triplets in *Protocol 1*, the CSP will update these chain hashes in the corresponding records. Now since chain hash of every latest resource version written in the current time slot is created using data owner's secret key  $K_s$ , no user including the writer of the resource itself can modify the resource. With the reasoning as in *Protocol 1*, in *Protocol 2* since MAC is computed over latest version' chain hash using data owner's key, the revoked user cannot be able to modify its own written records. The above two protocols are secure even if the unauthorized users will collude (*Definition 1*) because the secret key ( $K_s$ ) used in the computation of MAC is only known to the data owner. If the chain hash in last versioned record of a resource in a time slot is not MAC'ed with  $K_s$ , the CSP is caught by data owner for violating the mutual service agreement.  $\square$

## 4.1 Auditing algorithms

At the end of each time slot, along with the *Protocol 1* initiated by CSP, the data owner will execute the auditing process in co-operation with the CSP. Since the auditing process is executed by the data owner, every proof message received by a user against the corresponding read or write access request must be sent to the data owner as and when received. These proofs are used by the data owner at auditing time to verify their respective claims. Older proofs (whose new versions are committed and verified) can be deleted by the data owner from its store if necessary.

In the auditing process, the data owner will execute Algorithm 5. In the algorithm, the data owner selects the resource identifiers with version numbers to be audited and send them as read request to the CSP. The CSP will return the corresponding records back to the data owner. The data owner will then verify the stored proof messages (collected from various users) against the corresponding received records. The verification process is as follows. The proof messages collected from various users are first grouped according to the corresponding resources. Similar proof messages or messages for which resource does not exist (possibly deleted) are discarded. Now each group is verified against their corresponding chain received from the CSP. For each proof message, the data owner will check it with the corresponding record in the chain. If all proof messages are correct, the algorithm returns "Success" else it returns "Failure". In case the audit process returns "Failure", the data owner can frame charges against the CSP. Since the proofs are not forgeable, the CSP cannot deny the charges.

For efficiency reasons, we assume that the data owner will divide the outsourced resources into four groups (1–4) according to

their security level (similar to the priorities assigned in [16]). Most secure resources are in group 1 and resources which require the least secrecy are in group 4. Group 1 resources are audited in each time slot, group 2 resources are audited very frequently but not necessarily in each time slot, group 3 are least frequently audited and group 4 resources are never audited. To isolate the scheduled versions for auditing from the CSP, resources from groups 2 and 3 are randomly chosen.

## 4.2 Performance analysis

We implemented the proposed write access control system on Microsoft's Windows Azure [7] cloud platform using Java. In the implementation, we are not focusing on Azure storage performance since the proposed system can be implemented on most of the cloud storage systems. The implementation, in particular, focuses on cryptographic models used in the system, write operation latency and performance of the data owner machine.

The implementation consists mainly of three entities: a client machine in user premises and two virtual machines. Since our focus is on the performance of the data owner machine, one virtual machine works as a data owner and other works as a cloud server. The single client machine is used to generate a set of user threads mimicking the number of users in the system (used for stress testing at data owner). The client machine consists of an Intel core 2 quad processor 2.66GHz with 3GB RAM and 16MB Cache. We used AES-128 as the cipher for file encryption and SHA1 as the hash function. The cloud server stores 1000 records whose size varies from 100KB to 200KB.

Figure 2 shows the *put* transaction latency chart with respect to a different number of write requests given to the cloud service provider in *Protocol 1*. The major part of the processing time is taken by to and fro communication between the client machine and the cloud server machine. The remaining time is due to the cryptographic operations performed at the server.

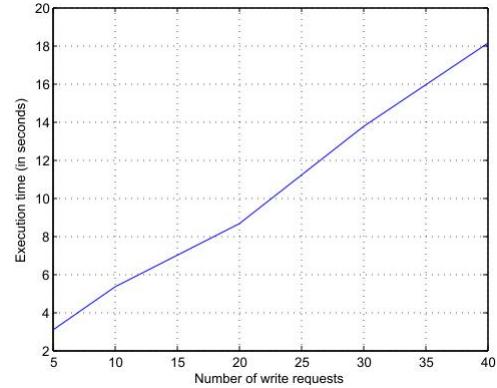


Figure 2: Put transaction latency

The performance of data owner machine for computing MACs over received message (containing chain hashes) in *Protocol 1* is at a rate of  $2.6 \times 10^4$  MB/s. For each received message, it includes computation of one MAC for checking received message integrity, verification of authorization certificate and MAC computations over received chain hashes (in the message) from the CSP.

Figure 3 gives the results for *Protocol 2*. The protocol requires two communications between the data owner and the CSP. In the first communication, the data owner sends a list of users IDs to the CSP whose write access has been revoked and receives a list of the corresponding chain hashes from the CSP. The data owner then computes the MAC over each received chain hash and sends

them back to the CSP, who will then update each corresponding record and return an acknowledgment back to the data owner. In the implementation, we consider three sets of revoked users i.e., 10, 100 and 1000. For each set 10 and 100 number of MACs are computed by the data owner, respectively. The maximum time in the operation is due to communication between the two parties. It may be noted here that in general the number of revoked users in a time slot and their respective latest committed versions (in the last time slot) will be limited in number.

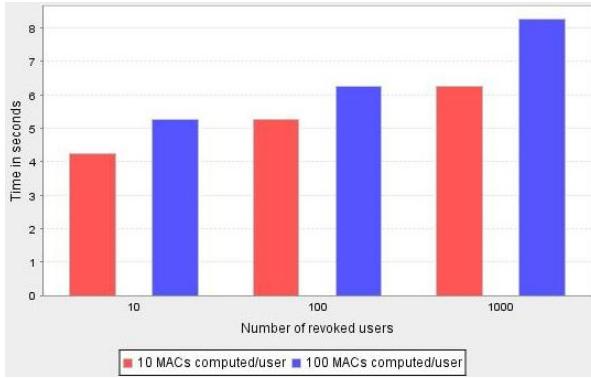


Figure 3: Implementation cost for *Protocol 2*

## 5. IMPROVING FRESHNESS

To request read access for a resource in Popa et al. [16] scheme, a user sends the resource ID along with a (fresh) random nonce to the CSP. The CSP authenticates the user and returns the requested resource information along with a proof message containing the nonce. The presence of nonce in the proof message only assures that the message is computed afresh. In [10, 8], a timestamp is stored securely with each outsourced record. The timestamp captures the time when the record was written.

In previously proposed schemes ([10, 16, 8]), the CSP can give stale resource in response to a read request without the misbehavior being detected by the data owner. The reason is that all of the schemes do not guarantee that there is any record written between the time when the returned record was written and the time when the resource is sent to the user. Therefore, we expect that there must be a commitment message that will guarantee “the returned resource version will be the latest committed version at least till the time when the data record is sent by the CSP to a reader”. In our model of distributed cloud servers, we need to give a practical interpretation to the staleness definition. A genuine issue is that there are delays in updating replicas; therefore one possible excuse from the CSP for such delays is that the requested server is disconnected (for a time being) from other servers. Now, the challenge is can we give a better/improved staleness (or freshness) guarantee?

### 5.1 New staleness notion

In order to guarantee the freshness of the received data to a user, in an ideal situation, the CSP needs to give a freshness proof (along with the data item) that will ensure the user that received data is fresh as the last update. However, there is a delay involved in sending and receiving the proof message. The existing notion of  $\langle k, t \rangle$ -staleness [3] ensures that a read request which begins  $t$  time units after the write commit operation returns one of the last  $k$  updated versions. We propose an alternate staleness

definition as follows.

**DEFINITION 2** ( $\langle v, d, t \rangle$ -STALENESS). *A system guarantees  $\langle v, d, t \rangle$  staleness if it assures that the version  $v$  of resource  $d$  is the latest committed version at least until time  $t$ , where  $t$  is the time when the resource was send by the CSP to a reader.*

The  $\langle v, d, t \rangle$ -staleness definition says that its respective freshness proof will ensure that the data version (i.e.,  $v$ ) returned is fresh as the last update at least till the proof creation time (i.e.,  $t$ ). In distributed scenario, no server can give such a proof since the write operation on a resource can be performed at any server. To protect against stale data, staleness sensitive applications require such concrete proofs so that if the CSP misbehaves then it must be caught by the data owner during the auditing process. Therefore in order to give a better interpretation to the freshness requirement, we constraint the proposed system so that such strong proof of freshness (providing  $\langle v, d, t \rangle$ -staleness) can be given.

In our proposed solution, we assume there is a dedicated write server for each resource i.e., in the system’s view, each resource is updated at only one dedicated server by the CSP and replicas are then updated. This idea is similar to the one used by Zellag et al. [18] and Akal et al. [1] where all resources are stored at a central database server. Every resource is first written at the central server then replicas are updated. On the other hand, in our model, there are more than one such dedicated servers each one is responsible for updating a distinct set of resources. It may happen that the CSP may violate the write updates on dedicated servers. However, they need to provide freshness guarantees in the form of proofs to the readers whenever asked. In this way, if the CSP does not follow the required agreement, it will be caught by the data owner during the auditing process with the help of proof messages (collected at the data owner from various users). It is to be noted here that the use of dedicated server may degrade the *get* and *put* transactions costs due to the increase in communication cost.

Therefore, the dedicated server can only generate proofs for their associated resources since other servers may not have yet received the latest updates. Upon a read request, the proof message is computed as follows.

$$\text{proof message} = \{h(t_{curr}, \text{resource\_info}, t_w)\}_{Pr_{CSP}}$$

Where,  $t_{curr}$  is the current timestamp indicating the time when the proof was created,  $t_w$  is the timestamp indicating the time when the resource was written, and  $\text{resource\_info}$  contains the  $t_w$  along with other resource record information. The proof message is sent to the read authorized user along with  $t_{curr}$  and resource information. It will assert to the read user that there is no record written in-between the proof creation time and the time when the returned resource was written. If later any message was found written in this time interval, the proof message can be used as a proof of staleness.

A system that ensures  $\langle v, d, t \rangle$ -staleness is called  $\langle v, d, t \rangle$ -staleness secure. An informal proof for the staleness secure property is given below.

**CLAIM 2.** *Proposed mechanism is  $\langle v, d, t \rangle$ -staleness secure.*

**PROOF.** *The proof message by the CSP binds the committed time  $t_w$  of the resource version along with the time  $t_{curr}$  when the proof is created. Such proofs are collected at the data owner and are used while auditing process. Since the proof message was signed by the CSP with its private key and created by a single dedicated server, it cannot later deny about the commitment given in the proof. In case*

the server tries to fool a reader by giving a stale data, the corresponding proof messages collected from authorized users must differ.

Suppose the version number of received resource record is  $v_i$  and its commit timestamp is  $t$ . The data owner (while auditing process) will check the timestamp ( $t'$ ) of version  $v_{i+1}$  (if exists). If  $t' < t_{curr}$  (present in the proof message), the record was stale else it was fresh. In case  $v_{i+1}$  does not exist, the record is assumed to be fresh. Since the timestamps in data records are used in computing MAC with writer's secret key, CSP cannot forge it. Hence, no unauthorized user (including the CSP) can give the stale record to an authorized user without being detected by the data owner.  $\square$

## 6. CONCLUSIONS

We discussed the challenges with regard to write access control that arises out of dependency on the untrusted service provider. We argued that even the authorized user should not be allowed to modify an outsourced content for an unlimited amount of time. Furthermore, a revoked user in collusion with the CSP should not be allowed to modify the old written data files for which his access is now revoked. We proposed two audit-based protocols to assure the data owner of the absence of unauthorized writes.

## 7. REFERENCES

- [1] F. Akal, C. Türker, H. Schek, Y. Breitbart, T. Grabs, and L. Veen. Fine-grained replication and scheduling with freshness and correctness guarantees. In *Proc. of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2*, pages 565–576, 2005.
- [2] M. Arapinis, S. Bursuc, and M. Ryan. Privacy-supporting cloud computing by in-browser key translation. *Journal of Computer Security*, 21(6):847–880, 2013.
- [3] P. Bailis, S. Venkataraman, M. J. Franklin, J. M. Hellerstein, and I. Stoica. Probabilistically bounded staleness for practical partial quorums. *PVLDB*, 5(8):776–787, 2012.
- [4] A. F. Barsoum and M. A. Hasan. Enabling dynamic data and indirect mutual trust for cloud computing storage systems. *IEEE Trans. Parallel Distrib. Syst.*, 24(12):2375–2385, 2013.
- [5] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conf., Santa Barbara, California, USA, August 18-22*, pages 1–15, 1996.
- [6] G. Cattaneo, L. Catuogno, A. D. Sorbo, and P. Persiano. The design and implementation of a transparent cryptographic file system for UNIX. In *Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference, June 25-30, Boston, Massachusetts, USA*, pages 199–212, 2001.
- [7] M. Corporation. Windows azure.  
<http://www.microsoft.com/windowsazure>.
- [8] S. D. C. di Vimercati, S. Foresti, S. Jajodia, G. Livraga, S. Paraboschi, and P. Samarati. Enforcing dynamic write privileges in data outsourcing. *Computers & Security*, 39:47–63, 2013.
- [9] K. E. Fu. Group sharing and random access in cryptographic file systems. *Master's thesis, MIT*, 1999.
- [10] E. Goh, H. Shacham, N. Modadugu, and D. Boneh. Sirius: Securing remote untrusted storage. In *Proc. of the Network and Distributed Sys. Security Symposium, NDSS 2003, San Diego, California, USA*, 2003.
- [11] M. Golfarelli, J. Lechtenbörger, S. Rizzi, and G. Vossen. Schema versioning in data warehouses. In *Conceptual Modeling for Advanced Application Domains, ER 2004 Workshops CoMoGIS, COMWIM, ECDM, CoMoA, DGOV, and ECOMO, Shanghai, China, November 8-12*, pages 415–428, 2004.
- [12] A. Juels and B. S. K. Jr. Pors: proofs of retrievability for large files. In *Proceedings of the 2007 ACM Conf. on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31*, pages 584–597, 2007.
- [13] J. Li, M. N. Krohn, D. Mazières, and D. Shasha. Secure untrusted data repository (SUNDR). In *6th Symp. on Operating System Design and Imple. (OSDI 2004), San Francisco, California, USA, December 6-8*, pages 121–136, 2004.
- [14] D. Mazières and D. Shasha. Building secure file systems out of byzantine storage. In *Proc. of the 21st Annual ACM Symposium on Principles of Distributed Computing, PODC 2002, Monterey, California, USA, July 21-24*, pages 108–117, 2002.
- [15] E. L. Miller, D. D. E. Long, W. E. Freeman, and B. C. Reed. Strong security for distributed file systems. In *Proceedings of the 20th IEEE International Performance, Computing and Communications Conference (IPCCC '01)*, pages 34–40, 2001.
- [16] R. A. Popa, J. R. Lorch, D. Molnar, H. J., and L. Zhuang. Enabling security in cloud storage services with cloudproof. In *2011 USENIX Annual Technical Conference, Portland, OR, USA, June 15-17*, 2011.
- [17] A. Shraer, C. Cachin, A. Cidon, I. Keidar, Y. Michalevsky, and D. Shaket. Venus: verification for untrusted cloud storage. In *Proc. of the 2nd ACM Cloud Computing Security Workshop, CCSW 2010, Chicago, IL, USA, October 8*, pages 19–30, 2010.
- [18] K. Zellag and B. Kemme. How consistent is your cloud application? In *ACM Symposium on Cloud Computing, SOCC'12, San Jose, CA, USA, October 14-17*, pages 6:1–6:14, 2012.

# An Architecture for SDN Based Sensor Networks

Roy Friedman   David Sainz  
Computer Science Department  
Technion - Israel Institute of Technology  
Haifa 32000, Israel

Email:[{roy,dsainz}@cs.technion.ac.il](mailto:{roy,dsainz}@cs.technion.ac.il)

## ABSTRACT

*Software Defined Networks* (SDN) are considered an efficient and flexible network architecture for managing large scale networks, in particular as it decouple between hardware and software and facilitates global network optimizations. Incorporating SDN concepts into *Wireless Sensor Networks* (WSN) opens interesting research challenges. This paper proposes an architecture for applying SDN to WSN. The architecture relies on recent extendibility features of OpenFlow to enable reconfiguring sensor behavior and network forwarding tables according to changes such as arrival of new sensors or change in user requests, as well as to apply dynamic in-network aggregation and filtering options. An evaluation of our approach for the specific case of in-network aggregation and its beneficial impact on the number of sent messages is also presented.

## CCS Concepts

- Networks → Network architectures; Network services; Ad hoc networks;

## Keywords

Software Defined Networks, SDN, Wireless Sensor Networks, WSN, OpenFlow, Open vSwitch

## 1. INTRODUCTION

Wireless Sensor Networks (WSN) enable deploying large numbers of typically cheap and low powered sensors in various terrains, while still being able to communicate with these sensors and obtain their readings at far away locations. This is largely due to their wireless device-to-device forwarding capabilities, which enable such low power communication over an extended range. In particular, WSN play a central role in the Internet of Things (IoT) and smart cities visions.

The constant improvements in hardware, software and communication have opened new ways to solve various problems inherent to this kind of networks. One of these advance-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

*ICDCN '17, January 04 - 07, 2017, Hyderabad, India*

© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007758>

ments is Software Defined Network (SDN) [5], an efficient and flexible network architecture for managing large scale networks. A key architectural concept behind SDN is that the control and data planes should be decoupled [18]. The data handling rules are not embedded in hardware, but are rather written as a software module in the control plane. This way the network is much easier to control by administrators. Changes to the control plane are easier and more global, without a per-device configuration of each hardware unit [12]. One of the most used standards for SDN is OpenFlow [2, 39], a protocol that implements the SDN concept in both hardware and software.

Software Defined Sensor Network (SDSN) [43] emerges as a way to use SDN in Sensor Networks, creating software-defined sensors that can change its sensing behavior, connectivity and deployment dynamically by loading different programs on demand. Yet, practical realizations of this concept are rare, and in particular, applying OpenFlow as the main vehicle for implementing SDSN involves non-trivial challenges.

We propose an architecture for WSN that enables SDSN based on OpenFlow, which can be re-programmable, aiming for an efficient decoupling between hardware and software. In this network, sensors are dynamic devices that may change their behavior according to environmental changes, demand and supply level of certain sensed data or user requests. The architecture proposed allows reconfiguring sensors' behavior and the network structure according to changes in the surroundings, such as arrival of new sensors or change in user requests.

The rest of this paper is organized as follows: Section 2 surveys the related work. Section 3 briefly describes the benefits applying SDN to WSN. Section 4 explains the conceptual design of our architecture and its additions to the OpenFlow protocol. Section 5 describes the implementation and testing. We conclude in Section 6 where we also elicit some future work.

## 2. RELATED WORK

Wireless Sensor Networks have been widely researched for the past years and it is still an ongoing research topic [6, 11, 34]. From routing and communication protocols [21, 27, 33, 40, 42] that tackle the challenges of these networks, to deployments for real applications [10, 32] or security mechanisms and protocols [17, 35], this field has experienced great advancements.

SDN is an active research field [5, 8, 18], which has sparked interest in both industry and academia, and has advanced

enough to have its own standard and implementations [25, 39].

Software Defined Sensor Networks have been recently explored as a result of applying SDN to sensor networks. In [43], the authors propose SDN to implement the routing mechanisms of mesh networks. The authors in [41] propose an architecture based on SDN to manage the network behavior of the Internet of Things (IoT). In [14] and [30], the authors propose SDN as a way to tackle inherent problems of WSNs such as difficulty to manage policy changes. They elaborate a theoretical analysis of how SDN integration could help and which changes can be made in the network, mainly using the OpenFlow standard applied to routing and communication mechanisms. In [23], the authors apply SDN to Wireless Sensor Networks to create a routing mechanism prototype implemented using the OpenFlow standard. They offer an evaluation based on a simulation program. In [43], the authors discuss an initial theoretical architecture to enable SDSNs, which is also cloud-enabled. They propose the idea of *sensing as a service* and leverage SDN for the network configuration. These efforts offer initial solutions for SDSNs. But, to the best of our knowledge, there are no implementations available and most of the work is theoretical.

In [20] the authors define a stateful architecture based on SDN for sensor networks that preserves some status information between packets and enables aggregation. However, this architecture is seen as a non-standard module on top of SDN. Even though it preserves concepts like flow tables, it deviates from many SDN concepts and creates a separate architecture. Thus, it does not offer modifications to the OpenFlow standard or known controllers, and rather uses modules created specifically for their system. In contrast, our architecture is designed to be accommodated in a standard OpenFlow switch, with the minimal modifications of the OpenFlow protocol we describe. One of the goals of our architecture is to describe the extensions to OpenFlow necessary for a standard OpenFlow switch to be used for sensor networks.

Reprogramming sensor networks is not a novel idea. Previous work before the arrival of SDN offers ways of reprogramming the networks using different techniques [34], including full firmware image reprogramming [26] and virtual machines [29]. The idea of programming or updating sensor firmware without manually changing every device was explored with Over The air Programming (OAP) [22], which states a way of programming sensors through commands and data transferred over the network. In [38], the authors propose a protocol for code dissemination over network of sensors, though they assume some kind of Over the Air Programming mechanism that is not provided. The authors in [15] propose a secure way of code dissemination over the air, proposing schemes that are less CPU-intensive and enabling the reprogramming of sensors, avoiding forgery of firmware images or viruses.

Besides code dissemination, sensor network frameworks that enable reprogramming of code have been explored. Some works like the operating system Contiki [16] support dynamic modules, which can be loaded at run time. An operating system able to load and unload modules or a framework able to load programs dynamically over a rigid operating system is a fundamental support for SDSNs [43]. Still SDSN aims for a higher decoupling level between the operating system and the programs and control layer. Aguilla [19] and

SensorWare [9] are frameworks implemented on top of sensor network operating systems. They are based on mobile agents and scripts that can replicate themselves over the the sensor network, making the behavior of the sensors dynamic and re-programmable. There are no mechanisms to delete scripts and it is also difficult to fully reprogram the behavior of the entire network. It would also be desirable to have a pull mechanism where the sensor asks for new behavior and pulls a new program from a source (even the cloud) given certain conditions.

### 3. WSN BASED ON SDN

#### 3.1 General Concepts

Our architecture is largely based on the conventional SDN concepts and the OpenFlow standard, adapted to the specific settings of WSN. In OpenFlow there are 2 main planes of action: the *data plane* and the *control plane*. In the data plane, devices are called *switches* and packets are organized as *flows*, each of which can contain multiple data packets. The data plane manages the *flow tables*, which hold the information about how to redirect and manipulate the data flows.

The table is divided into flow entries, which have 2 main fields. The *match field* contains the conditions a packet needs to comply in order to be managed and the *action field* specifies the action to execute. The most common actions are to discard the packet, to forward it or to encapsulate it and send it to the control plane.

When a packet arrives to a switch, it is matched against the match fields of the table to check if there is an entry that satisfies the conditions. If such an entry exists, the switch executes the proper action specified in the flow entry. If none of the entries satisfy the conditions, the packet is encapsulated and sent to the control plane to receive instructions on how to manipulate it. With those instructions, new flow entries are added to the table so the switch knows how to manipulate similar packets in the future. The basic idea is that SDN switches forward packets based on this table, without a general knowledge of the routing policies in the network. That part, or basically the logic for managing the network, is implemented separately in the control plane.

In the control plane, devices are called *controllers*, and they hold a general vision of the network. One of the most common controllers implementation is NOX [1], written in C++. These controllers are in charge of the routing rules and the topology of the network, and instruct the switches in the data plane to modify their flow tables accordingly. The communication between the controller and the switch is done via the OpenFlow protocol, normally through a reliable and secure transport mechanism. The existing OpenFlow standard is primarily designed for a unitary controller holding the whole network vision in a centralized manner [25]. The last draft of the standard at the time of this research is OpenFlow 5.0, though its implementation in software and devices is not fully accomplished yet.

#### 3.2 Benefits of Applying SDN to WSN

Standard WSNs are typically composed of sensors that are independent and able to carry tasks for themselves, from the hardware layer up to the application layer. They are made compact and often have their software tightly coupled with their hardware specifications. The reason is to maximize ef-

iciency and battery life. Due to the nature of WSNs, many sensors need to be deployed away from power grids and are required to have battery independence. Having such specific software specifications often means proprietary, vendor-specific software that renders sensors inflexible and hard to program.

This problem becomes more profound if different sensors from various vendors are deployed in the WSN, which is normally the case. The network soon needs to be maintained using different vendor-specific software versions. The network becomes very difficult to maintain and deploying a new version of the software becomes increasingly complicated. Many times, changes need to be made manually in sensors.

New versions of software may be required as well, not only for new developments but also for changes in details like routing policies or configuration-driven behavior changes.

Applying the SDN principles to WSNs brings multiple advantages:

- A unified way of programming sensor behavior: Different sensors (possibly from different vendors) that adopt SDN become more easily programmable using a single unified paradigm: flow tables for the flow of data in the sensors, a common layer for central controlling in the control plane and a standard way of communication between controller and sensors via the OpenFlow standard. This paradigm greatly reduces the software fragmentation typical of WSN, offering a more seamless way of programming.
- Easier policy changes: Sensors can change their behavior adding policy changes simply by making changes in their flow tables. This change can be made from the control plane, in a global and consistent manner throughout the network and without changing any software in the sensors.
- A unified ground for application creation: Developing new applications for WSN is a matter of building on top of the common control plane, which offers known APIs and coding rules.

## 4. DESIGN

The architecture that we designed applies the SDN principles to WSN. In our architecture, nodes are seen as devices interconnected to each other. Nodes are configured inside a network topology and are classified into different categories. The *Controller node* holds the vision of the network and the routing policies; this node carries the functions proper of an SDN controller. *Intermediate nodes* are nodes with enough resources to add some logic and follow the routing policies given by the controller. These nodes carry the functions proper of the SDN switches and hold the flow tables. *Leaf nodes* are simple nodes with not enough resources to implement extra logic, and behave like classic nodes of a WSN, connected to intermediate nodes and serving as information sources for them.

Intermediate nodes form the data plane of the SDN while the controller node forms the control plane. Note that a node can be at the same time an intermediate node and a leaf node, holding the data flow logic while reading from its sensors and creating the data packets, creating a software defined sensor or flow sensor [31].

When sensors read values, the information is sent through the network until the destination point, usually the controller node. They go through intermediate nodes that route the packets via the flow tables. The policies are easily maintained and changed through the controller node, reaching the intermediate nodes that adapt their flow tables accordingly. This separates the tasks of data transmission carried by intermediate nodes from the data control carried by the controller node, and through a standard implementation.

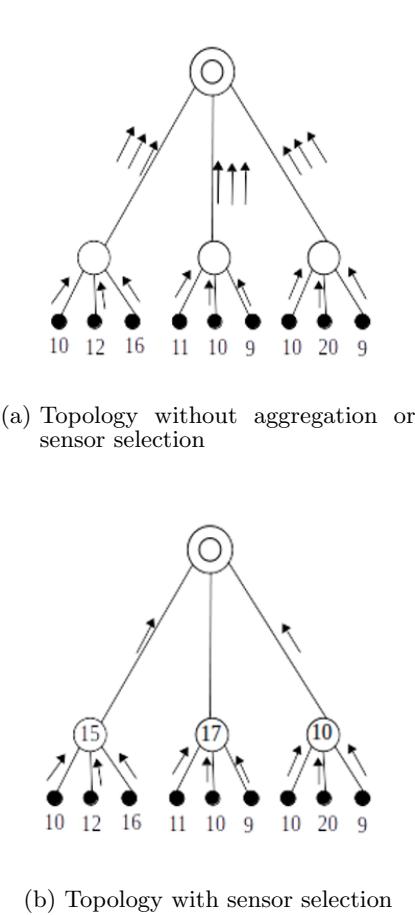
Intermediate nodes can reach the controller via OpenFlow for control messages. However, it is possible for a node to have a different routing path for the sensor information and the OpenFlow messages. It is possible that messages from sensors pass through several intermediate nodes through different routes before reaching the controller node. As an example, it is possible that an intermediate node has a network interface, used to communicate with the controller node, that consumes a lot of energy. At the same time, it has a secondary interface used for the sensor messages that consumes much less energy but has limited range. Thus, messages need to be routed through other intermediate nodes to reach the destination. The energy-consuming interface is used only for the less frequent case of communication with the controller.

### 4.1 Using SDN to Further Program Sensor Behavior

Besides routing mechanisms, SDN can be used to manage other aspects inherent to sensor networks. Here, we exploit additional OpenFlow features to tackle problems such as *sensor selection* and *data aggregation*.

In SDN, the flow tables are assumed to use some kind of IP-like addressing, which is not generally the case in WSNs. In sensor networks, it is possible to address and select sensors by type or value range, regardless of the concrete sensor node. In many cases, it is necessary not only to receive values of a concrete type, but also the values that belong to a concrete value range, discarding the rest. In order to achieve this kind of sensor addressing and selection, flow tables need to be adapted for creating new types of matches that take into account these differences. By allowing sensor types and value ranges in the flow tables, we encapsulate part of the basic execution also in the SDN. Changing value ranges or types of sensors to listen to becomes a task of the controller node, which for example would instruct an intermediate node to change the flow tables so a packet is forwarded only if the value is a temperature read and it surpasses a given threshold. Some of the basic logic in WSNs can be controlled by flow tables, and the sensor reconfiguration can still be achieved in the control plane by OpenFlow controller messages. By addressing specific sensors and adding thresholds to values, the architecture is able to reduce sent messages, forwarding only the relevant sensor values.

Data aggregation is a sensor network technique in which *aggregator* nodes are appointed. These nodes have the task of receiving messages from different sensors and apply an aggregation function to them, such as averaging. The aggregator node can send only one aggregated message instead of each of the individual messages received. Intermediate nodes can acquire this aggregation task. In order to achieve data aggregation, flow tables need a mechanism capable of making small computations on simple values coming from sen-



**Figure 1: Example of message reduction achieved with sensor selection and aggregation.**

sors. They also need to be capable of storing the computed value. This mechanism adds extra work to the data plane, but it has valuable advantages. Furthermore, switches already perform these kind of computations and value storage in the data plane for features like flow statistics. As values arrive to the intermediate nodes, the aggregation is computed and stored. Note that aggregation can be combined with selection using thresholds, this time for aggregated values. Besides the advantages mentioned in Section 3.2, there is another benefit from using SDN: by executing the aggregation and selection in the data plane instead of the control plane or some other central node, the system can discard messages in early stages, before forwarding them further, which greatly reduces the number of messages sent in the system. For example, the message could forward only the average of the sensor values when it surpasses a threshold.

These modifications make it possible for switches to perform certain small tasks, while not straying from the OpenFlow standard. Ultimately, the controller is in charge of defining the type of sensors to listen to, the ranges of the values and the aggregation functions to perform. All the logic is still decided in the control plane while the data plane still handles the flow data without general knowledge of the network.

HEADER						
OXM_CLASS	OXM_FIELD	HM	OXM_LENGTH	OXM_VALUE	OXM_MASK (OPTIONAL)	
HEADER						
NAME	WIDTH	USAGE				
OXM_TYPE	OXM_CLASS	16	Match class: member class or reserved class			
	OXM_FIELD	7	Match field within the class			
	HM	1	Has mask: set if OXM include a <del>bitmask</del> in payload			
	OXM_LENGTH	8	Length of OXM payload			
OXM_VALUE						
CAV_OFFSET	CAV_CAST	CAV_OP	CAV_VALUE			
NAME	WIDTH	USAGE				
CAV_OFFSET	16	Starting position of the attribute				
CAV_CAST	4	Data type of the attribute				
CAV_OP	4	Operator				
CAV_VALUE	Determined by CAV_CAST	Value to use the operator with				

**Figure 2: Format of the CAV OXM type.**

Figure 1 illustrates an example of message reduction achieved with sensor selection. Figure 1(a) shows one network configuration and the flow of messages through it without sensor selection or data aggregation. Figure 1(b) shows the same network configuration but with sensor selection enabled. In this figure, we can see how nodes are arranged in a tree. The top node is the controller node, and the target of messages. White nodes are intermediate nodes and the black nodes placed on the leaves of the tree are low nodes. In the example, sensors values are read in the low nodes, which send messages that are routed through the intermediate nodes (using the flow tables) until they reach the controller node. The objective of the controller is to get the maximum value from all sensors.

In Figure 1(a), it can be seen that all messages need to be sent to the controller so it can make the calculations and thresholds. A total of 18 messages are sent. In Figure 1(b), intermediate nodes have selection active for only values higher than a specific number. The figure shows how the intermediate nodes store the threshold. Messages pass through the network topology and are processed in the intermediate nodes, then passed through if the selection criteria is satisfied. A total of only 11 messages are sent.

Adding aggregation mechanisms imply calculation and writing operations in the switches, or intermediate nodes. An OpenFlow switch already performs these kind of calculations for various purposes, e.g, the calculation of statistics in flow entries. Changing values in the packets is also a common task already, since there are numerous actions, like COPY\_FIELD, meant to change values in packet headers, etc. The writing operations required for aggregation are also tasks that OpenFlow switches can already perform.

## 4.2 Sensor Addressing and Selection

In order to achieve sensor addressing and selection, the

TABLE 1		TABLE 2	
Match (OXM)	Instruction	Match (OXM)	Instruction
Type=Temperature	GOTO_TABLE 2	Temperature<25	Action: DROP

OXM for table 1:				
OXM_TYPE ...	CAV_OFFSET	CAV_CAST	CAV_OP	CAV_VALUE
OXM_CAV	50	int16	"="	1(temperature)
OXM for table 2:				
OXM_TYPE ...	CAV_OFFSET	CAV_CAST	CAV_OP	CAV_VALUE
OXM_CAV	52	int32	"<"	25

**Figure 3: One example of the CAV OXM type in flow tables.**

architecture implements a solution similar to the idea suggested in [30] that exploits the OpenFlow extensible match (OXM), a type length value (TLV) format used to express matches in the flow tables. We define a new match type by adding an OXM class with attributes to specify the type of sensor and the value ranges of interest. It basically points at the sensor value of interest and gives a value and an operator to compare it to. The format of the new OXM class called *CAV* is shown in Figure 2. The first field tells the offset where the value to compare starts. This is typically the value of the sensor read in the packet. The next field determines the type of value and also its length. The CAV\_OP field determines the operator to use in the comparison. Finally, the CAV\_VALUE field holds the value.

As an example, a switch can receive information from several types of sensors. It can be configured to receive specific sensor types through concrete ports, so the switch always knows what type of value to expect. Alternatively, the sensors can send along the value the type of sensor it belongs to. Figure 3 shows an example of the latter. The packet is matched against the first flow table, which looks into the type of value and directs the packet through the pipeline. In the example, the packet is directed to table 2 to treat temperature values.

The packet is then matched against table 2, where the figure shows a further example of sensor selection. The packet is discarded since the value is less than 25. Note that in the example it is also possible to concatenate the 2 CAV values in a single flow table match, and have one flow table instead of a pipeline.

### 4.3 Data Aggregation

The data aggregation that we handle in the data plane consists of fairly simple calculations like averaging or calculating the maximum between 2 values. Typically, these functions require a value derived from previous packets (as the last calculated average or the current maximum). This would normally imply keeping a state between packets in a flow, which SDN switches do not support at the moment. Instead, they support different per-flow counters used for statistics. In our architecture, the aggregation calculations are seen as new statistics added to the flow entry, using a specific register to store values like other statistics do. That way, the aggregated value of the flow lasts across different packets. The new statistics are recalculated and stored when new packets arrive and match a specific flow entry. Until

OpenFlow 1.5, the statistics had a much more rigid format and were not extendible. Version 1.5 takes care of this issue and introduces the idea of extensible statistics, allowing new types of statistics to be supported and calculated.

The controller could periodically ask for the statistics of the tables and collect them. However, this approach suffers from several drawbacks:

- Extra messages: Polling generates messages for the repetition, and it might be necessary to poll several times before obtaining a new value. This also generates multiple messages for the response.
- Energy: Extra messages cost extra energy to send and receive.
- Time: Polling in large intervals may result in lost values and may require too much time to obtain a read, which does not fit the usual real-time constraints of these networks. Polling in small intervals may incur a prohibitively high message overhead.

Thus, our architecture adopts a message push approach. The aggregated values are sent to the target once they are updated. There are two possible different ways of supporting message push:

- Use of triggers: Version 1.5 of OpenFlow adds a trigger functionality to the statistics, where messages containing statistics can be pushed under certain events.
- Copy push: Upon a flow match, the aggregation is calculated and written in a register. In subsequent flow matches in the pipeline, the aggregated value can be checked accordingly to perform selection.

We opted to implement the approach using copy push, as it is more fine-grained and more useful in scenarios where the aggregated value cannot be sent using the OpenFlow protocol or the controller channel consumes too much energy. Moreover, the control channel may be busy with control messages to comply with real-time constraints for sensor data. The copy push approach also enables *nested aggregation*: value aggregation from data that has been previously aggregated by another intermediate node. Aggregated values can traverse different intermediate nodes and be treated as normal sensor values, instead of traveling across the network encapsulated in an OpenFlow message through the controller channel.

In order for the aggregated values to be pushed toward the destination without using a controller channel, they need to travel in a packet, not only be stored as statistics in the switch. We opted for writing the aggregated value in the same packet as the sensor data, either overwriting the previous sensor value or writing it in a different offset.

The aggregation function performs this task itself. If it is not done by the aggregation function, extra packet fields (possibly dedicated) would be necessary. It would also incur additional write operations and the modification of the COPY\_FIELD instruction, to be able to perform the copy of the value as part of the slower instruction set of the flow entry.

In order to represent aggregated values in OpenFlow, we use the OpenFlow extensible statistics (OXS) format, a type length value format used to express statistics in the flow tables. The implementation of the new statistics for aggregation uses the following new fields:

HEADER				
OXS_CLASS	OXS_FIELD	RE	OXS_LENGTH	OXS_VALUE
<b>HEADER</b>				
NAME	WIDTH	USAGE		
OXS_TYPE	16	Stat class: member class or reserved class		
OXS_FIELD	7	Stat field within the class		
RE	1	Reserved for future use		
OXS_LENGTH	8	Length of OXS payload		
<b>OXS_VALUE</b>				
AGG_FUNCTION	HV	AGG_CASTS	AGG_OFFSETS	AGG_CASTD
AGG_OFFSETED				AGG_OFFSETD
AGG_VALUE	Determined by AGG_CASTD			AGG_VALUE (OPTIONAL)
NAME	WIDTH	USAGE		
AGG_FUNCTION	7	Function used for aggregation		
HV	1	Set if OXS_AGG includes the aggregated value.		
AGG_CASTS	4	Data type of the source aggregation value		
AGG_OFFSETS	16	Starting position of the source attribute to aggregate		
AGG_CASTD	4	Data type of the destination aggregation value		
AGG_OFFSETED	16	Starting position of the destination attribute to write		
AGG_VALUE	Determined by AGG_CASTD	Aggregated value		

Figure 4: Format of the new OXS\_AGG type used for aggregation in OpenFlow.

- Accumulator registers: Used to store the aggregated value. If the aggregation function needs more than one register to perform the aggregation, there will be extra accumulator registers.
- AGG\_FUNCTION: A field dictating which kind of aggregation function is applied (maximum, minimum, average, etc.).
- AGG\_CASTS: A field with data type of the source value.
- AGG\_CASTD: A field containing the data type of the aggregated value.
- AGG\_OFFSETS: A field with the packet offset where the source value starts.
- AGG\_OFFSETED: A field with the packet offset where the aggregation value should be written. The offset can be the same as the source offset, meaning overwriting the value.

Figure 4 shows the new OXS type named OXS\_AGG that is used to describe the desired aggregation in different OpenFlow messages and instructions. It holds all the aforementioned fields. However, the OXS representation only needs one accumulator field to store the aggregated value. The rest of the accumulators are seen as helpers and are not necessary to pass in messages. The OXS representation also holds one more field determining whether the aggregated value is included.

In order to properly set the aggregation in the flow tables, a modification in the OpenFlow protocol is needed. When the controller sends a message to add a flow entry in the switch, it can specify an OXS\_AGG field to define which type

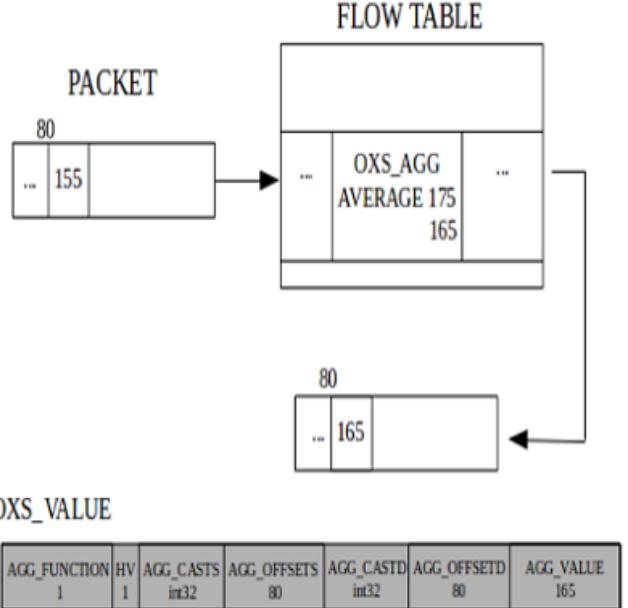


Figure 5: Example of a OXS\_AGG type holding a integer average function.

of aggregation is wanted. If no field is sent, the aggregation will not take place. The message is sent with OXS\_HV set to zero accordingly. The switch will also need to add its support to OXS\_AGG when describing its features.

Once a specific aggregation is set in a flow entry, packets that match the entry trigger the calculation of the aggregation, as it triggers the calculation of the rest of the statistics regarding that flow. The accumulator is modified according to the selected aggregation function and so is the packet.

Figure 5 shows an example of an OXS\_AGG aggregation in a flow table. The function calculates the average of integers. The packet matches the flow shown in the table and hence the calculation is performed, passing the resulting value to the packet. The values of the registers necessary for this example of aggregation are represented in OXS\_AGG format at the bottom of the figure.

## 5. IMPLEMENTATION AND TESTING

For the implementation, we modified Open vSwitch [3] for the SDN switch. Our contribution resides largely in the switch, where we added the new registers and operations. Open vSwitch as of today, still did not catch up with OpenFlow version 1.4, so some of the new 1.5 capabilities needed to be implemented as well.

We modified the switch to accommodate the changes. With the use and evolution of OVS to more modern OpenFlow versions, we hope to achieve a more refined modification to the switch. Open vSwitch uses in Linux a kernel module to handle the packets, while a program in user space runs the OpenFlow module and keeps the flow tables. There is a representation of the flow tables in the kernel, though for efficiency reasons it differs considerably from the OpenFlow pipeline held in user space [36]. This change prevents some action combinations and certain pipelines are restricted.

**Table 1: Configuration of the parameters of the test for the random traces.**

$D_1, D_2$	-10,100
Increment $d$	$\text{sqrt}(D_2 - D_1)/2$
Max nodes connected to a node	4
Messages sent per sensor	500

For sensor selection, we added extra fields to the implementation, e.g., for an offset of the CAV value, and created the new OXM value. For sensor aggregation, per-flow registers are added and the aggregation function is applied as an internal action.

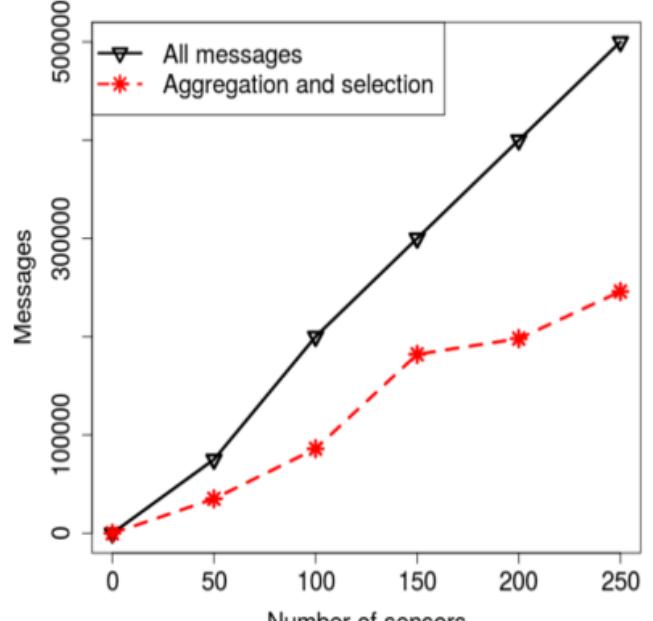
After modifying the switch, we created a network environment to test its functionality and behavior. We created different networks using the Mininet environment [13], varying the number of nodes to reflect the impact in terms of message saving. We used a hierarchical topology, as it is the most common topology for large wireless sensor networks [28]. Low nodes gather sensor information and report to their respective intermediate node, which performs aggregation and selection calculations and then reports to the main, root node. We also used Wireshark [4] to monitor network traffic.

First, we evaluated our architecture with a random sensor value trace. The scenarios with this trace are configured as follows: In each scenario, low nodes collect integer values in the range  $[D_1, D_2]$  at a constant interval and in subsequent reads, they yield a new value obtained by applying a random increment or decrement  $d$  to the prior value. Intermediate nodes are designated at random either an aggregation function or a threshold (with a random value in the range  $(D_1, D_2)$ ), under which the value is not sent further. Each low node sends the same number of messages with sensor values. We counted the packets with sensor/aggregation values.

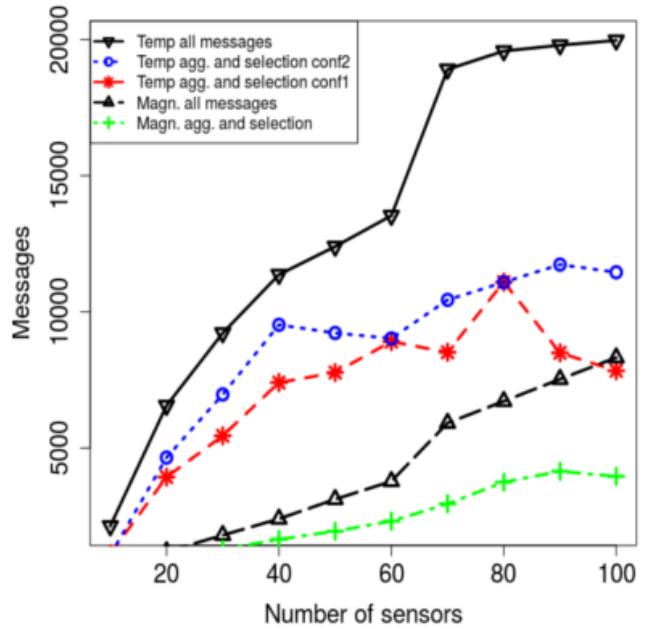
Each scenario has a different number of nodes in the network, and was run 50 times, averaging the results. Table 1 reflects the parameter configuration of the test.

Figure 6(a) shows the different scenarios executed with this trace file, with their number of nodes and messages. The continuous line shows the number of messages without sensor selection, which are proportional to  $\log_{f_o}(\text{nodes})$ , where  $f_o$  is the maximum number of nodes connected to a node. Consequently, its progression is more linear. The dotted line shows the number of messages once the aggregation and selection are applied. The message reduction is substantial. Due to the characteristics of these scenarios, a stable message reduction ratio is expected. However, different real life scenarios can have characteristics and configurations with a much more dramatic message reduction, e.g., if the target is only interested in values that greatly exceed the average or are atypically low. Nevertheless, the test was also useful to demonstrate the viability of an implementation of our architecture design in a widely-used switch.

In order to evaluate the architecture in more realistic cases, we also used different real traces from sensors. First, we used temperature measurements in the city of Rome [7]. In this evaluation, each node of our network acts as a temperature sensor of an area of the city (a total of 100) and emits temperature reads along with a timestamp. We cre-



(a) Evaluation of the random traces



(b) Evaluation of the temperature and magnetometer sensor traces

**Figure 6: Evaluations made with different sensor traces**

ated two tests with this trace file. The first test, named *conf1*, has the switches configured in the same manner as the random trace test. The second test, named *conf2*, has the switches with threshold configured in the mean value obtained from the traces, instead of a threshold chosen at random. We also used traces from a magnetometer [37] placed in different locations, from which we randomly selected 100. The switch configuration is similar to the random trace test. Each location outputs around 20 magnetometer reads and constitutes a node. These tests evaluate scenarios with different number of nodes. Figure 6(b) shows the message reduction achieved with these trace files. In both tests, we can observe a substantial message reduction as well.

This message reduction impacts directly on the energy consumption. As stated in [24], a node consumes energy for both sending and receiving messages, and it depends on the number of bits sent, the energy to run one bit through the specific circuitry and the distance that the signal amplifier tries to achieve. The obtained energy reduction is crucial in most WSN environments and greatly improves the lifespan of the nodes.

## 6. CONCLUSION AND FUTURE WORK

In this paper, we have studied the utility and feasibility of applying SDN concepts to wireless sensor networks. Specifically, to motivate this study, we have started by exploring the advantages of applying SDN to wireless sensor networks, such as the ease of management of routing policies in flow tables. We have also studied the benefits and practicality of adding more functionality proper to these networks in the intermediate nodes, such as applying simple aggregation capabilities, while adhering to the principles of SDN and recent additions to the OpenFlow standard. Next, we have presented an architecture that enables wireless sensor networks to benefit from SDN’s concepts and existing infrastructure. We have explained how to utilize these new features of the OpenFlow protocol to define corresponding extensions in order to accommodate the desired capabilities of sensor selection and data aggregation, which we have implemented in a real OpenFlow switch. Finally, we have experimented with our augmented implementation in various scenarios through emulation, which has exemplified significant message reduction when applying such sensor selection and aggregation at intermediate nodes.

Looking into the future, we intend to explore additional extensions that allow new selection functions for value changes. We are also considering implementing the OpenFlow statistics triggers for aggregation in order to enable more possibilities. As Open vSwitch evolves, accommodating our modifications should become even more streamlined, as upcoming features in its architecture become available.

**Acknowledgements.** This work was partially funded by the Israeli Ministry of Science and Technology, grant #3-10886.

## 7. REFERENCES

- [1] Nox controller implementation.  
<http://www.noxrepo.org/nox/about-nox/>.
- [2] Open flow architecture. <http://www.openflow.org/>.
- [3] Open vswitch switch implementation.  
<http://http://openvswitch.org/>.
- [4] Wireshark network analyzer.  
<https://www.wireshark.org/>.
- [5] AGARWAL, S., KODIALAM, M., AND LAKSHMAN, T. Traffic engineering in software defined networks. In *INFOCOM, 2013 Proceedings IEEE* (2013), IEEE, pp. 2211–2219.
- [6] AKYILDIZ, I. F., SU, W., SANKARASUBRAMANIAM, Y., AND CAYIRCI, E. A survey on sensor networks. *Communications magazine, IEEE* 40, 8 (2002), 102–114.
- [7] ALSWAILIM, M. A., HASSANEIN, H. S., AND ZULKERNINE, M. CRAWDAD dataset queensu/crowd\_temperature (v. 2015-11-20), nov 2015.
- [8] BAKSHI, K. Considerations for software defined networking (sdn): Approaches and use cases. In *Aerospace Conference, 2013 IEEE* (2013), IEEE, pp. 1–9.
- [9] BOULIS, A., HAN, C.-C., AND SRIVASTAVA, M. B. Design and implementation of a framework for efficient and programmable sensor networks. In *Proceedings of the 1st international conference on Mobile systems, applications and services* (2003), ACM, pp. 187–200.
- [10] BUTUN, I., MORGERA, S. D., AND SANKAR, R. A survey of intrusion detection systems in wireless sensor networks. *Communications Surveys & Tutorials, IEEE* 16, 1 (2014), 266–282.
- [11] CUZZOCREA, A., FORTINO, G., AND RANA, O. Managing data and processes in cloud-enabled large-scale sensor networks: state-of-the-art and future research directions. In *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on* (2013), IEEE, pp. 583–588.
- [12] DAS, S., PARULKAR, G., McKEOWN, N., SINGH, P., GETACHEW, D., AND ONG, L. Packet and circuit network convergence with openflow. In *Optical Fiber Communication Conference* (2010), Optical Society of America, p. OTuG1.
- [13] DE OLIVEIRA, R. L. S., SHINODA, A. A., SCHWEITZER, C. M., AND RODRIGUES PRETE, L. Using mininet for emulation and prototyping software-defined networks. In *Communications and Computing (COLCOM), 2014 IEEE Colombian Conference on* (2014), IEEE, pp. 1–6.
- [14] DELY, P., KASSLER, A., AND BAYER, N. Openflow for wireless mesh networks. In *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on* (2011), IEEE, pp. 1–6.
- [15] DENG, J., HAN, R., AND MISHRA, S. Secure code distribution in dynamically programmable wireless sensor networks. In *Proceedings of the 5th international conference on Information processing in sensor networks* (2006), ACM, pp. 292–300.
- [16] DUNKELS, A., GRONVALL, B., AND VOIGT, T. Contiki-a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on* (2004), IEEE, pp. 455–462.
- [17] ESCHENAUER, L., AND GLIGOR, V. D. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM conference on Computer and communications security* (2002),

- ACM, pp. 41–47.
- [18] FANG, S., YU, Y., FOH, C. H., AND AUNG, K. M. M. A loss-free multipathing solution for data center network using software-defined networking approach. In *APMRC, 2012 Digest* (2012), IEEE, pp. 1–8.
- [19] FOK, C.-L., ROMAN, G.-C., AND LU, C. Rapid development and flexible deployment of adaptive wireless sensor network applications. In *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on* (2005), IEEE, pp. 653–662.
- [20] GALLUCCIO, L., MILARDO, S., MORABITO, G., AND PALAZZO, S. Sdn-wise: Design, prototyping and experimentation of a stateful sdn solution for wireless sensor networks. In *2015 IEEE Conference on Computer Communications (INFOCOM)* (2015), IEEE, pp. 513–521.
- [21] GUO, S., HE, L., GU, Y., JIANG, B., AND HE, T. Opportunistic flooding in low-duty-cycle wireless sensor networks with unreliable links. *Computers, IEEE Transactions on* 63, 11 (2014), 2787–2802.
- [22] HAGEDORN, A., STAROBINSKI, D., AND TRACHTENBERG, A. Rateless deluge: Over-the-air programming of wireless sensor networks using random linear codes. In *Proceedings of the 7th international conference on Information processing in sensor networks* (2008), IEEE Computer Society, pp. 457–466.
- [23] HAN, Z.-J., AND REN, W. A novel wireless sensor networks structure based on the sdn. *International Journal of Distributed Sensor Networks* 2014 (2014).
- [24] HEINZELMAN, W. R., CHANDRAKASAN, A., AND BALAKRISHNAN, H. Energy-efficient communication protocol for wireless microsensor networks. In *System sciences, 2000. Proceedings of the 33rd annual Hawaii international conference on* (2000), IEEE, pp. 10–pp.
- [25] HU, F., HAO, Q., AND BAO, K. A survey on software defined networking (sdn) and openflow: From concept to implementation.
- [26] HUI, J. W., AND CULLER, D. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proceedings of the 2nd international conference on Embedded networked sensor systems* (2004), ACM, pp. 81–94.
- [27] INTANAGONWIWAT, C., GOVINDAN, R., AND ESTRIN, D. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking* (2000), ACM, pp. 56–67.
- [28] KAUR, G., AND GARG, R. M. Energy efficient topologies for wireless sensor networks. *International Journal of Distributed and Parallel Systems (IJDPS)* 3 (2012), 179–192.
- [29] LEVIS, P., AND CULLER, D. Maté: A tiny virtual machine for sensor networks. In *ACM Sigplan Notices* (2002), vol. 37, ACM, pp. 85–95.
- [30] LUO, T., TAN, H.-P., AND QUEK, T. Q. Sensor openflow: Enabling software-defined wireless sensor networks. *Communications Letters, IEEE* 16, 11 (2012), 1896–1899.
- [31] MAHMUD, A., RAHMANI, R., AND KANTER, T. Deployment of flow-sensors in internet of things' virtualization via openflow. In *Mobile, Ubiquitous, and Intelligent Computing (MUSIC), 2012 Third FTRA International Conference on* (2012), IEEE, pp. 195–200.
- [32] MAINWARING, A., CULLER, D., POLASTRE, J., SZEWCZYK, R., AND ANDERSON, J. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications* (2002), ACM, pp. 88–97.
- [33] MANJESHWAR, A., AND AGRAWAL, D. P. Teen: a routing protocol for enhanced efficiency in wireless sensor networks. In *Parallel and Distributed Processing Symposium, International* (2001), vol. 3, IEEE Computer Society, pp. 30189a–30189a.
- [34] MOTTOLA, L., AND PICCO, G. P. Programming wireless sensor networks: Fundamental concepts and state of the art. *ACM Computing Surveys (CSUR)* 43, 3 (2011), 19.
- [35] PERRIG, A., SZEWCZYK, R., TYGAR, J., WEN, V., AND CULLER, D. E. Spins: Security protocols for sensor networks. *Wireless networks* 8, 5 (2002), 521–534.
- [36] PFAFF, B., PETTIT, J., KOPONEN, T., JACKSON, E., ZHOU, A., RAJAHALME, J., GROSS, J., WANG, A., STRINGER, J., SHELAR, P., ET AL. The design and implementation of open vswitch. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)* (2015), pp. 117–130.
- [37] PUROHIT, A., PAN, S., CHEN, K., SUN, Z., AND ZHANG, P. CRAWDAD dataset cmu/supermarket (v. 2014-05-27). Downloaded from <http://crawdad.org/cmu/supermarket/20140527>, may 2014.
- [38] REIJERS, N., AND LANGENDOEN, K. Efficient code distribution in wireless sensor networks. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications* (2003), ACM, pp. 60–67.
- [39] SHERWOOD, R., CHAN, M., COVINGTON, A., GIBB, G., FLAJSLIK, M., HANDIGOL, N., HUANG, T.-Y., KAZEMIAN, P., KOBAYASHI, M., NAOUS, J., ET AL. Carving research slices out of your production networks with openflow. *ACM SIGCOMM Computer Communication Review* 40, 1 (2010), 129–130.
- [40] TUNCA, C., ISIK, S., DONMEZ, M. Y., AND ERSOY, C. Distributed mobile sink routing for wireless sensor networks: A survey. *Communications Surveys & Tutorials, IEEE* 16, 2 (2014), 877–897.
- [41] WU, D., ARKHIPOV, D. I., ASMARE, E., QIN, Z., AND MCCANN, J. A. Ubiflow: Mobility management in urban-scale software defined iot. In *2015 IEEE Conference on Computer Communications (INFOCOM)* (2015), IEEE, pp. 208–216.
- [42] YOUNIS, M., SENTURK, I. F., AKKAYA, K., LEE, S., AND SENEL, F. Topology management techniques for tolerating node failures in wireless sensor networks: A survey. *Computer Networks* 58 (2014), 254–283.
- [43] ZENG, D., MIYAZAKI, T., GUO, S., TSUKAHARA, T., KITAMICHI, J., AND HAYASHI, T. Evolution of software-defined sensor networks. In *Mobile Ad-hoc and Sensor Networks (MSN), 2013 IEEE Ninth*

*International Conference on* (2013), IEEE,  
pp. 410–413.

# FastRank: Practical Lightweight Tolerance to Rational Behaviour in Edge Assisted Streaming

Xavier Vilaça, Luís Rodrigues, João Silva  
 INESC-ID, Instituto Superior Técnico  
 Universidade de Lisboa  
 {xavier.vilaca,ler,joao.roque}@tecnico.ulisboa.pt

Hugo Miranda  
 LASIGE, Faculdade de Ciências  
 Universidade de Lisboa  
 hamiranda@ciencias.ulisboa.pt

## ABSTRACT

Edge-computing is one of the most promising techniques to leverage the excess capacity that exists at users' premises. Unfortunately, edge-computing may be vulnerable to free-riding, i.e., to nodes that attempt to benefit from the system without providing any service in return. Traditional approaches model free-riders as rational nodes that strive to maximize a utility and apply Game Theory concepts to devise mechanisms that deny any utility gain to nodes that deviate from the protocol. These mechanisms impose significant overheads. This paper proposes a new approach that avoids these overheads, which applies concepts of evolutionary game theory. We propose to devise lightweight mechanisms targeted for the optimistic setting where the vast majority of nodes adopts one of a small number of behaviours. More precisely, we assume that most nodes are altruistic or follow non-sophisticated behaviours such as free-riding or white-washing. If a small fraction of nodes follows alternative behaviours, then our lightweight mechanism limits the utility gain of these nodes, making it unlikely that the number of nodes exploiting sophisticated behaviours increases at a fast pace. This allows altruistic nodes to detect their presence in time to switch to more robust mechanisms, before the system reaches a state where the lightweight mechanisms can no longer cope with the existing behaviours. We apply this approach in the context of edge-assisted streaming.

## 1. INTRODUCTION

Decentralised peer-to-peer systems are a powerful tool to explore the unused capacity at the edges of the network. Unfortunately, it has been observed [1, 3, 10] that a significant fraction of nodes may free-ride by not contributing to the system, while still benefiting from the cooperation of a sufficiently large fraction of nodes that cooperate unconditionally, known as altruistic nodes. This not only puts an unfair load on altruistic nodes, but also degrades the performance of the tasks executed at the edge. Mechanisms that can detect free-riders and prevent them from dominating the

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

*ICDCN '17, January 04-07, 2017, Hyderabad, India*

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007750>

system operation are therefore extremely relevant.

Existing work [4, 7, 14, 15] has addressed free-riding by adopting a game theoretical approach. In this approach, nodes are assumed to be rational, i.e., nodes do not contribute because they aim at maximising a utility, which decreases with the amount of resources provided to other nodes and increases with the benefits of participating in the system. These works devised incentive mechanisms that are incentive-compatible, i.e., they deny any utility gain to nodes that deviate from the protocol. This approach has two main drawbacks. First, because rational nodes can deviate from the protocol in an arbitrary fashion, sophisticated incentives are necessary, which are generally costly. Second, nodes may not be capable of determining which of two protocols provides a higher utility.

Evolutionary Game Theory (EGT) is an alternative framework to Game Theory that does not assume that nodes are fully rational [8, 20]. Instead, every node  $i$  starts by following some "hardwired" protocol. At each point in time,  $i$  may change to a different protocol with some probability, either by replicating the behaviour of other nodes or by mutating to a new protocol. The replication probability is proportional to the utility difference between protocols. The aim is to devise protocols that are *stable* in the sense that, if all nodes are following the protocol and a small fraction of nodes mutates into following another protocol, then this fraction does not tend to grow with time. This models the realistic scenarios where nodes start by following some known pre-specified protocol (e.g., BitTorrent in file-sharing), but then some of them may discover alternative protocols that they believe to provide a higher utility. Subsequently, other nodes may be influenced by them and also change their behaviour.

In this paper, we advocate an adaptive approach inspired in the concepts of EGT to devise efficient and stable incentive mechanisms. The approach comprises at least two different protocols: one is a lightweight protocol that is stable in an optimistic setting where nodes may deviate from the protocol in a small number of ways, whereas the other is a more costly stable protocol for arbitrary settings. While we can find in the literature several (costly) protocols that address arbitrary settings (such as [4, 7, 14, 15]), to the best of our knowledge, there are no examples of lightweight protocols that are both efficient and stable in an optimistic setting. This paper bridges this gap by proposing such a protocol, named FastRank.

Evidence collected from past and current usage of edge-assisted systems, including peer-to-peer file sharing and live-streaming systems, confirms that in most decentralized sys-

tems, most users either volunteers to provide service to others (i.e., are altruistic) or deviate from the protocol by adopting simple free-riding or white-washing behaviours [3, 19, 10]. We argue that the reason why more sophisticated behaviours are not frequently observed is because they require developing an alternative protocol, which most users are not willing or capable to do. Recent experiments have shown that a streaming system cannot provide high reliability if a majority of users deviates [19]. This motivates us to target the lightweight protocol to an optimistic setting with at most 30% users that free-ride or white-wash. This is intended to capture an initial setting where most users have not yet discovered the benefits of deviating and thus have not mutated into following more beneficial protocols. If a higher fraction of deviating users is expected in the initial setting, then the system designer should require users to follow a more costly and robust protocol from the beginning. To make the system stable to mutations, the lightweight protocol must provide a metric sensible to non-expected behaviours (i.e., different from free-riding and white-washing), so that if altruistic nodes detect that the fraction of nodes deviating in an unexpected way is too large, then they may switch to a more robust protocol. The lightweight protocol must also limit the utility gain of nodes with more sophisticated behaviour to limit the growth rate of the fraction of these nodes until the switch operation is concluded.

To demonstrate the applicability of our approach, we address the problem of edge-assisted streaming. The usefulness of edge-computing to support live streaming has been demonstrated by several large scale real-life deployments, including PPLive[9] among others[16]. We propose a lightweight protocol that is stable in a setting with altruists, free-riders, and white-washers. Like most live-streaming implementations, we require nodes to join an overlay network in order to receive and forward packets of the stream. We consider that free-riders never forward packets, and white-washers strive to circumvent the incentive mechanisms of our protocol by constantly acquiring new neighbours from whom they receive the stream. Our overlay maintenance protocols include incentives for nodes to keep stable symmetric links with a small set of neighbours. As a result, it becomes possible to use simple direct reciprocity mechanisms to deny them the benefit of receiving the stream. We also limit the frequency with which nodes can create new relationships to cope with white-washing behaviour. Based on these principles, we propose FastRank, an integrated topology management and peer-monitoring scheme that can effectively and efficiently minimize the impact of free-riders and white-washers. Interestingly, we also show that the approach limits the utility gain of more sophisticated behaviours. Furthermore, we show that, if the fraction of nodes that follow sophisticated behaviours grows to a point where it impacts the reliability, this can be detected by altruistic nodes that can then trigger a system reconfiguration and commute to a more robust protocol.

FastRank has been extensively evaluated. Our experimental results show that FastRank is able to keep altruistic nodes connected to each other, while isolating free-riders, denying them access to the stream, and limiting the frequency with which white-washers create new relationships to the point where they do not gain from this strategy. We also show that these incentives are sufficient to limit the utility gain of nodes adopting a more sophisticated behaviour

receive a limited benefit and that, if their fraction is large, their presence can be detected by altruistic nodes in time to switch to a more robust protocol.

The rest of the paper is structured as follows. Section 2 surveys related work. Section 3 presents the model. Sections 4 and 5 describe FastRank and how it can be leveraged to build a full adaptive solution. Section 6 reports on our experimental results. Finally, Section 7 concludes the paper.

## 2. RELATED WORK

There are two main approaches to support live-streaming in large scale systems: one is to build multicast dissemination trees and the other is to rely on gossip protocols. In optimal conditions, i.e. with minimal churn and a negligible number of free-riders, tree based approaches are a natural and efficient structure to spread information as they avoid any redundant message delivery. However, in an uncontrolled environment like the Internet, the performance of tree-based approaches degrades rapidly because tree need to be reconstructed whenever a node disconnects or does not conform with the protocol. Gossip protocols do not suffer from this limitation.

A gossip dissemination protocol can be decomposed in two components. The membership component is responsible for the definition and maintenance of the overlay topology. Challenges are in the definition of a topology that is simultaneously resilient, in order to mitigate the effects of churn, and efficient, presenting a small signalling cost even for a large number of participants. Two protocols that address this problem are SCAMP [6] and HyParView [13]. The second component of gossip is the dissemination strategy, which makes use of the overlay network to disseminate data. Message dissemination typically follows an epidemic style data dissemination, with nodes working in rounds, and selecting *fanout* nodes to relay data items in each round. This data distribution approach has shown to be effective and resilient to failures as long as non-failed nodes follow the protocol. However, the redundancy of gossip does not apply if nodes do not follow the protocol and avoid forwarding messages.

To mitigate the impact of nodes that do not follow the protocol, BAR Gossip [15] relies on a Balanced Exchange mechanism. In an interaction between two nodes  $n_1$  and  $n_2$ , data relayed from  $n_1$  to  $n_2$  is encrypted before being sent and  $n_1$  only releases the encryption key after receiving a comparable amount of data from  $n_2$ . The Balanced Exchange mechanism can be seen as a stronger implementation of the tit-for-tat approach that can be found in Bit-Torrent [3]. Although efficient, encryption imposes a considerable overhead, which is amplified in live content streaming due to the large number of packets exchanged and their timely delivery requirement.

FlightPath [14] contributes to attenuate the overhead of the brute-force approach proposed by BAR gossip in two ways. On one hand, it uses a flow control mechanism, where nodes negotiate in advance the exchange of packets. Flow control permits to balance evenly the traffic on each round and decrease redundancy. Secondly, FlightPath creates a balance account between pairs of nodes, which substitutes BAR Gossip's by a more efficient mechanism where some nodes may fall behind others although within previously agreed thresholds.

LiFTinG [7] implements a distributed mechanism for detecting free-riding in asymmetric environments, i.e. when it

is expected that a node sends more data to another node than it receives from him. LiFTinG requires nodes to track others nodes behaviour by cross-checking the history of their previous interactions. The randomness of node selection plays a key role in LiFTing by preventing colluding behaviour, where nodes can choose to send information always to the same subset of third parties, possibly uncovering a group’s misbehaviour. However, cross-checking information incurs a non-negligible overhead.

FullReview [4] proposes an alternative mechanism for detecting free-riding. Nodes keep tamper-proof logs of all the received and sent messages. A log of a node  $i$  allows for other nodes to verify whether  $i$  followed the protocol. Periodically, a set of monitors of  $i$  requests and verifies  $i$ ’s log, and triggers a punishment of  $i$  in case a deviation is detected. Unfortunately, this approach incurs the significant overheads of periodic logging and log transfer.

### 3. SYSTEM MODEL

We focus on the problem of disseminating a stream in a gossip fashion from a source to all the nodes of a peer-to-peer overlay network. The source first partitions the stream into multiple frames and sends them to a subset of nodes. Then nodes cooperate to disseminate the stream by forwarding each received frame to a subset of nodes selected at random. Each node  $n$  has an identifier and a *view* of the system, i.e., a subset of identifiers of other nodes to which  $n$  forwards frames. The nodes from the view of  $n$  are called the neighbours of  $n$ . The number of neighbours that  $n$  has in its view is the out-degree of  $n$ , and the number of neighbours that have  $n$  in their views is the in-degree of  $n$ . A view of  $n$  is symmetric if every node in that view has  $n$  in its view.

Our model of behaviours is based on models of EGT [8, 20]. We assume that the source is trusted and never fails. Nodes may fail by crashing and their behaviour up to the point where they crash is determined by their *utility* and *strategy*. The utility is difference between the benefit of receiving the stream and the communication costs of executing the protocol. The benefit of receiving the stream increases with the amount of frames received by the node, and the costs increase with the number of sent messages. Strategies correspond to protocols executed by the nodes. We characterize the set of different strategies according to (1) the *forwarding strategy*, specifying a probability distribution over the nodes from the view of  $n$  to which  $n$  forwards each previously received frame from the stream, (2) a *view strategy*, specifying the set of nodes that  $n$  keeps in its view, and (3) an *identity strategy*, specifying the set of different identities that  $n$  uses to participate in the system. The forwarding strategy directly affects the utility of node  $n$  in terms of the communication costs of forwarding frames of the stream. The view and identity strategies of  $n$  affects the interactions between  $n$  and other nodes, thus, affecting the number of potential nodes that forward frames to  $n$  and the frames sent by  $n$ . We target our protocol to an optimistic setting with four types of nodes that follow different strategies: *altruistic*, *free-riders*, *white-washers*, and *hybrid*. Altruistic nodes follow the specified protocol. Free-riders never forward frames of the stream, but adopt the identity and view strategies specified by the protocol. White-washers deviate from the protocol in the view and identity strategies by trying to acquire more neighbours than what is specified by the protocol. Hybrid nodes adopt a combination of

free-riding and white-washing strategies.

Models of EGT consider dynamic populations, where the fraction of nodes following each strategy may vary. Normally, this variation is captured by the replicator dynamics rule. According to this rule, the fraction of nodes following a given strategy increases at a rate proportional to the gain in the utility of following this strategy. Given this, a protocol is *stable* if, in a setting where a majority of nodes follow the protocol, there is no utility gain from deviating from the protocol, such that the fraction of nodes following a different protocol does not increase. Our aim is to devise a protocol that is stable in a setting with a majority of altruists, but where a fraction of nodes may follow free-riding and white-washing strategies.

## 4. FastRank

We now describe FastRank, a peer-to-peer streaming service that is stable in a setting with altruists, free-riders, and white-washers. FastRank has three main components: an overlay network construction and maintenance protocol, a localised neighbour ranking mechanism, and a dissemination mechanism.

### 4.1 Overlay Network

In order to build and maintain symmetric relationships, we use HyParView[13], which is a peer-to-peer protocol that constructs and maintains a symmetric overlay appropriate for dissemination. In this overlay, after establishing a relationship, two nodes preserve it until one of them fails. The number of neighbours of each node is deliberately small (i.e., logarithmic with regard to the system size) [12], such that neighbours are required to interact frequently during the message dissemination process. This is especially suited for nodes monitoring each other locally and detecting free-riding behaviour. Unfortunately, HyParView cannot be used as a black box, given that it has been designed under the assumption that all nodes are altruistic, and thus does not cope with free-riders and white-washers. FastRank extends HyParView by constraining the rate at which new relationships are created, and implements local monitoring based on a ranking mechanism.

**Constrained HyParView.** When a node  $i$  contacts another node  $j$ , to create a new relationship between  $i$  and  $j$ , node  $i$  is given a time-consuming task that it needs to perform before the relationship request is accepted. This, in practice, introduces a *quarantine period* before the establishment of a new relationship. The quarantine period should be long enough such that multiple frames are lost and white-washing becomes unappealing. Furthermore, the task given to node  $i$  should be such that no more than one task can be performed in parallel by a single node during a given quarantine period, i.e., if a node attempts to establish two new relationships, it should be forced to “pay” the cost of waiting two quarantine periods.

To achieve the goals above, in FastRank, we have opted to use crypto-puzzles. More precisely, when a node  $i$  contacts node  $j$  to establish a new relationship, node  $j$  prepares a crypto-puzzle that needs to be solved by  $i$  in order for  $j$  to accept the relationship. The crypto-puzzle is such that the estimated average time to solve it is the pre-defined quarantine period, even if the entire computing resources of a node are devoted to the task. Several examples of crypto-puzzles with these guarantees have been described in the

literature[21, 2]; in FastRank we have opted to use the hint-based hash-reversal scheme from [5]. A node joining the Constrained HyParView overlay contacts a target number of neighbours, gets a challenge from them, solves all the crypto-puzzles and, finally, provides the answers to all neighbours. In this way, it is likely that the joining node is accepted (approximately at the same time) by a number of neighbours that is large enough to initiate its operation without risking being marked as free-rider (because it does not receive enough information to forward).

**Active View Maintenance.** In HyParView a node proactively attempts to maintain his active view full. Therefore, if a neighbour crashes, it immediately attempts to establish a new relationship to refill its active view. However, with Constrained HyParView, actively attempting to establish a new relationship is costly. Furthermore, if very few nodes have empty slots in their active views, it is likely that multiple nodes concurrently compete for that entry (and only one will succeed). This further exacerbates the cost of joining an overlay where all nodes pro-actively attempt to fill their views as soon as possible. On the other hand, an altruistic node can opt to wait and refill its active view by accepting relationships from nodes attempting to join the network. If all altruistic nodes do this, not only they avoid the costs of initiating relationships but also they make the joining procedure easier for new altruistic nodes that want to be part of the overlay. In FastRank, we use a low watermark threshold, denoted *baseview*, that needs to be reached before the node pro-actively looks for neighbours. If some relationships end but the size of the active view is above *baseview*, the node simply waits for join requests, and will accept relationships until *maxview* is reached. This is a safety mechanism that ensures that newcomers have the possibility to join the system. When a node  $n$  has in its view *maxview* relationships and  $n$  receives a join request,  $n$  will request the new node to solve a crypto-puzzle. The first node to complete it will replace the node with the lowest rank in its view.

**Source Node.** In FastRank, the source of the stream is treated differently from every other node. The source does not keep an explicit active view to a fixed set of nodes. Instead, it uses a large passive view to select a number of contact points at random for each frame it sends. Nodes always receive frames sent directly by the source and altruistic nodes forward the frames to the neighbours in their active view. The goal is to distribute the load evenly among the members of the overlay, such that there is not a fixed set of nodes that is close to the source (and always has to forward frames) and another set of nodes that permanently act as leafs (and just receive frames).

**Passive View Maintenance.** In HyParView[13], every node maintains a random sample of the system membership called the *passive view*. The passive view is used to select nodes to establish the neighbourhood relations that define the active view. Random walks are used to periodically find new nodes and renew the passive view. In FastRank, instead of resorting to random walks, the passive view is provided by the source. This avoids the problem of nodes that manipulate the random walks to their own gain.

## 4.2 Ranking Algorithm

FastRank leverages the topological properties of Constrained HyParView to implement an efficient localised monitoring mechanism that can effectively detect and, ultimately, ex-

pel free-riders from the active view of altruistic nodes. The mechanism is based on the observation that, in steady relationships, two altruistic nodes roughly send the same amount of frames to each other. If the balance of exchanged frames is highly asymmetric, this is a sign that the node that is receiving but not forwarding frames is likely a free-rider or a failed node, and thus may be expelled from the view. The fact that Constrained HyParView keeps symmetric views plays a crucial role in this mechanism, since it ensures that altruistic nodes have a small set of neighbours with whom they interact repeatedly. This allows to detect any unbalance quickly.

The balance of the exchanges with a neighbour is captured by FastRank as a numeric rank[11]. Each node  $i$  maintains, for each neighbour  $j$  in its active view a separate  $rank_{ij}$ . The rank of a new neighbour is initiated to a predefined value, denoted the *baserank* and then maintained using a very simple rule that consists of incrementing the rank of the neighbour every time a frame is received from that neighbour and by decrementing the rank when a frame is sent to the neighbour. We take the value of *baserank* to be 0. This means that a neighbour that sends more frames than it receives keeps a positive score, and a free-rider will have negative score. Furthermore, the ranking algorithm also defines a minimum threshold for the rank, denoted *minrank*, below which a node is expelled from the view. In Section 6, we discuss how the value for *minrank* is selected.

## 4.3 Dissemination Mechanism

Frame dissemination is implemented as follows. The stream source selects, for each frame, a number  $f$  of contact points among the entire set of members of the overlay (this is achieved by keeping a large passive view) and then sends the frames to those nodes. When a node receives a frame for the first time (directly from the source or from one of its neighbours), the node forwards the frame to all neighbours in its active view, with a probability that is a function of the rank of that neighbour. The size of the active views of the overlay constructed by the Constrained HyParView are deliberately small but still large enough to tolerate a large fraction of faulty nodes. As a result, if all nodes are altruistic, the use of flooding in the overlay may generate many redundant messages. Therefore, an altruistic node forwards a message to other altruistic nodes with a probability lower than 1 that is called the *base forwarding probability* (or simply *bfp*). The value of *bfp* is selected such that the reliability of the dissemination is still ensured but with a much smaller cost than that incurred when flooding is used. In Section 6 we discuss how *bfp* can be configured for optimal results. We find that forwarding frames with fixed probability disregarding the ranks of the neighbours is not sufficient for our purposes. First, in the presence of heterogeneous nodes, a node will always send fewer frames than a faster neighbour, possibly being mistaken as a user that forwards frames with a lower probability. Second, before being expelled, free-riders may receive a large fraction of the stream. To address these problems, a node  $i$  forwards frames to a neighbour  $j$  with a probability  $fp_i(j)$  that varies proportionally to the rank of  $j$ : if the rank is higher than *baserank*, then  $i$  forwards frames with a probability higher than *bfp*, otherwise,  $fp_i(j)$  drops below *bfp*. This ensures that (1) nodes that forward frames with probabilities lower than *bfp* tend to receive less frames, thus decreasing their utility; and (2) a slower node

that falls behind in the number of frames sent in a relationship with another node  $j$  has the opportunity of raising its rank before the relationship is lost. Specifically, we use the following formula for  $fp_i(j)$ :

$$fp_i(j) = bfp \left| \frac{minrank - rank_{ij}}{minrank} \right|.$$

## 5. ADAPTIVE FRAMEWORK

FastRank provides a solution to streaming that is stable in a setting where non-altruistic nodes follow free-riding or white-washing strategies. However, this solution is not stable in general settings, where nodes may adopt other more sophisticated strategies, e.g., forwarding frames with a probability lower than what is specified by the protocol or varying the size of the active view. Given that FastRank is not stable in the general setting, if a fraction of nodes mutates into following more sophisticated strategies, then this fraction will grow without constraints. To tackle this issue, FastRank can be leveraged to build an adaptive framework that works as follows. Non-altruistic nodes probe the performance of the system according to some metric sensible to the presence of sophisticated behaviour (examples of metrics that can be used are the observed reliability or the observed latency, among others). Once this metric indicates the presence of a high fraction of nodes following sophisticated behaviours, altruistic nodes complain to the source. If the number of complaints raises above a certain threshold, then the source may disseminate an authenticated message that leads altruistic nodes to switch from FastRank to a more robust protocol  $P$ .

We argue that this adaptive solution is stable in general settings as long as the following requirements are met: (1)  $P$  must be stable for a fraction  $\alpha > 0$  of non-altruistic nodes following arbitrary strategies, (2) in FastRank, the utility gain of non-altruistic nodes must be sufficiently small, (3) the switching metric must be sufficiently sensible, so that, if an initial fraction of up to  $\alpha$  nodes are non-altruistic but only a very small fraction of nodes follows more sophisticated behaviours, then the fraction of non-altruistic nodes does not grow beyond  $\alpha$  prior to the switching operation, and (4) the switching metric should not be too sensible to the point of the probability of a false-positive switch caused by random fluctuations being non-negligible. Requirement (1) guarantees that, if the pre-condition for executing  $P$  is met at the time of the switch operation, then non-altruistic nodes will cease to obtain any utility gain, and their number will tend to diminish. (2) and (3) are necessary to satisfy the pre-condition of  $P$ . Finally, (4) ensures that the adaptive system will switch to  $P$  only if necessary, thus preserving our goal of maximizing efficiency in more optimistic settings. Experimental results presented in the evaluation section indeed suggest that FastRank allows to adjust the thresholds of the switch operation to satisfy requirements (2)-(4) simultaneously.

## 6. EVALUATION

In this section, we provide an extensive evaluation of FastRank using both simulations and experiments with a prototype on PlanetLab. We performed simulations using the PeerSim Framework [17]. Simulations consisted in the dissemination of 20000 frames among 1000 nodes, each injected

	value		value
<i>maxview</i>	15	<i>baseview</i>	12
<i>minrank</i>	-15	<i>bfp</i>	0.4
quarantine period (frames)			220

**Table 1: Default configuration of FastRank**

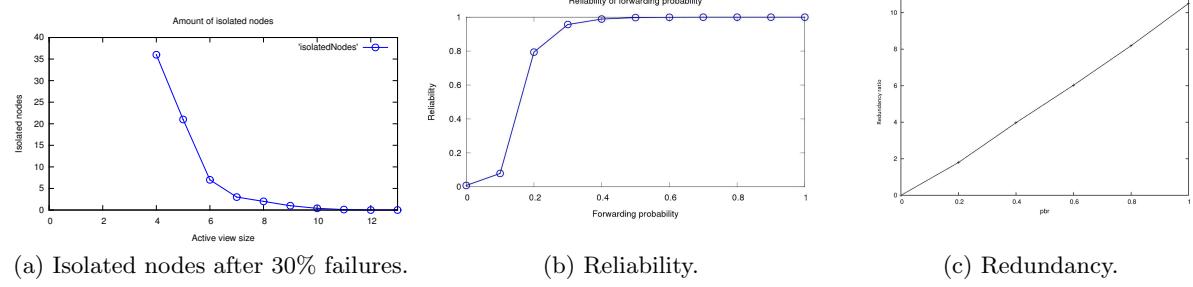
in the network by the streamer using 7 peers randomly chosen, at a rate of 100 kbps. Results presented in this section are the average of 100 independent runs using the same configuration. We performed an identical evaluation in a PlanetLab testbed with 16 machines. The evaluation is divided into seven different parts. The first five parts include results from our simulations, whereas the last two parts refers to experiments in PlanetLab. Namely, in the first part, we support the choice of values selected for the different parameters of the system. The second part illustrates the operation of the system when all nodes are altruistic. The third and fourth parts discuss the effects of free-riders and white-washers respectively. The fifth part analyses the effect of nodes that follow more sophisticated strategies. The sixth part provides an evaluation of the impact of heterogeneous nodes in an optimistic setting with free-riders and white-washers. The seventh part quantifies the overhead of FastRank compared to related systems.

### 6.1 Configuring FastRank

The parameters that affect the operation of FastRank are the following: the size of the active view and of *minview*; the parameter of the ranking procedure (*minrank*); the base forward probability *bfp* used in the dissemination process; and the average length of the *quarantine period* (i.e., the average time needed to solve the crypto-puzzle when creating a new relationship). Table 1 presents the default configuration values of FastRank. We discuss the rationale for configuration of these parameters in the following subsections. Unless stated otherwise, the discussion applies to a network of 1000 nodes, the setting used for the graphs depicted in the paper.

**View Size.** We first discuss how the size of the active view is selected. For now, let's assume that *maxview* is equal to *baseview* and that flooding is used to propagate the messages in the overlay. In the next section, we discuss why flooding is not used in FastRank. As long as the network of altruistic nodes remains connected, all correct nodes will receive all the messages. Therefore, the *baseview* size must be selected such that the likelihood of an altruistic node to become isolated in the presence of faulty nodes is very small. We have opted to configure FastRank such that at least 30% of faulty nodes can be tolerated with minimum effect on the altruistic nodes. Figure 1(a) shows the percentage of altruistic nodes that becomes isolated from the primary components of the overlay (the primary component is the largest connected subgraph in the overlay) for different sizes of the active view, after 30% of simultaneous failures. As it can be seen, if the *baseview* is equal or larger than 11, only 0.1% of altruistic nodes are isolated. Such a small value motivated us for using the conservative approach of selecting 12 as the default value for *baseview*. We have opted to add 3 additional slots to facilitate the inclusion of joining nodes, for a *maxview* size of 15.

**Forward Probability.** We now explain the rationale for selecting the message base fowarding probability *bfp*. In the



**Figure 1: Baseview and Forward Probability (reliability vs redundancy)**

spirit of gossip protocols, we avoid this redundancy by forwarding messages with a given probability smaller than 1. By fixing the *baseview* to 12, Figure 1(b) depicts the reliability of the streaming protocol on a FastRank overlay, as a function of the dissemination probability and Figure 1(c) depicts the resulting redundancy. By selecting a forward probability of 0.4 on an overlay where the *baseview* is 12, one can still achieve a very high reliability with a significant reduction in the redundancy of the dissemination procedure.

**Rank Mechanism.** The goal of the rank mechanism is to detect free-riders by equating a rank below *minrank* with free-riding behaviour. However, due to random fluctuations of dissemination, it is possible that the rank of altruistic nodes also drops below *minrank*, a situation that we call a *false positive*. Our goal is then to promptly detect free-riders while minimising false positives. Therefore, the value of *minrank* should weigh this trade-off. Figure 2(b) shows the fluctuation of the rank among two altruistic nodes. From these graphs it is clear that *minrank* should not be higher than -15. However, note that the lower the value of *minrank*, the less likely it is to generate a false positive but also the longer it would take to detect a free-rider, as shown in Figure 2(c). Since we aim at a fast detection of free riders, we have opted to use the maximum value that ensures a small fraction of false positives, i.e, the value of -15.

**Quarantine Time.** The goal of the quarantine time is to ensure that nodes cannot replace relationships faster than they are ended. From Figure 2(c) it can be observed that the last free-rider was detected after 110 frames for a *minrank* of -15. Therefore, the join procedure should use a crypto-puzzle that takes, on average, a time that is longer than the time it takes to forward that number of frames. Since we also aim at penalizing free-riders that also use a white-washing strategy, we have selected a quarantine period of twice the detection time (i.e, corresponding to 220 frames). In this way, free-riders that continuously attempt to replace old neighbours by new neighbours miss 50% of the frames.

## 6.2 Failure-free Altruistic Operation

In this subsection, we provide some additional insights on the operation of FastRank in a setting where all nodes are altruistic and do not fail. For this, we consider a stream of 24 frames per second, generated by one single stream source, during a complete session with 14 minutes. The session starts with 1000 nodes and in the middle of the stream (at minute 7), 100 additional nodes join the stream (i.e., at that momento we induce an 10% increase in the over-

Initial network size	1000
Joining nodes	100
False positives	0.03
Global reliability	0.999
Newcomers reliability	0.995
Newcomers average puzzles	12.2
Worst case time to join stream (frames)	3116

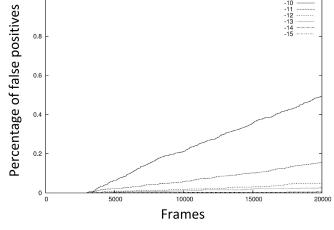
**Table 2: Operation in Absence of Misbehaviour**

lay population). With this setting, we show: the time it takes to setup the FastRank overlay, the reliability experienced by a node; the average number of retransmissions per frame received during the streaming session; the percentage of false-positives during the session; the amount of crypto-puzzles solve by each node during the session; and, finally, the time it takes for a new node to join an ongoing streaming session with 90% reliability.

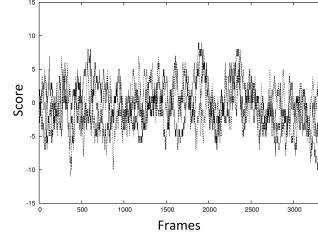
The results are depicted in Table 2. As it can be seen, the amount of crypto puzzles solved by joining members, 12.2, is slightly above the minimum (we note that a joining member must establish *baseview*, i.e., 12, neighbours). The difference is due to the contention for the free slots in the views of nodes that already belong to the overlay. But even in the worst case (for the last node to join) the time corresponds to solving 14 crypto puzzles, just 2 above the minimum. This contention is minimal and shows that having a *maxview* above *baseview* is quite effective at helping new members to join the overlay. It can also be observed that the joining of new members does not affect the reliability of the stream nor the accuracy of the free-rider detection mechanism.

## 6.3 Tolerating Free-Riders

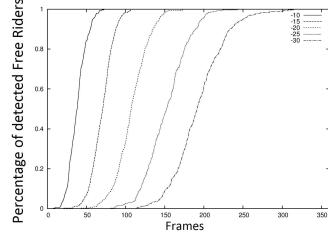
In this section, we provide some additional insights on the operation of FastRank in the presence of free-riders. In this experiments we let the system run with 100% altruistic nodes until a point where 30% of all nodes adopts the behaviour of a free-rider. Figure 3 shows the evolution of the composition of the active views of nodes after the injection of free-riding behaviour. The figure shows that after 110 frames, free-riders are detected and the system starts to reconfigure. After 2500 frames the system stabilizes in a configuration where 95% of the members in the active view of an altruistic node are other altruistic nodes. Consequently, free-riders have become disconnected from the network. We have also measured how many crypto-puzzles altruistic nodes and free-riders have solved during the recon-



(a) False positives.



(b) Rank fluctuation.



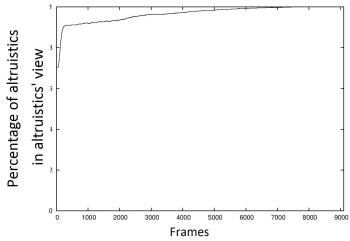
(c) Detection Time.

**Figure 2: Ranking**

figuration. Since the *baseview* is 12, with 30% free-riders, an altruistic node must, on average, replace 4 members in its view. However, in this experiment, an altruistic node has to solve 6.75 crypto-puzzles before it stabilizes its active view with other altruistic nodes, i.e., almost 3 more puzzles than in the ideal case. This is due to the fact that this experiments captures an extreme case, where all free-riders act simultaneously, and also keep continuously solving crypto-puzzles to replace their broken relationships, thus generating a significant contention for the free slots in the view of altruistic nodes.

#### 6.4 Tolerating White-Washers

In this subsection, we provide some additional insights on the operation of FastRank when a fraction of 30% nodes adopts a hybrid strategy of free-riding and white-washing, where the white-washing strategy consists in using an enlarged active view size. The main gain of this strategy may stem from avoiding the costs of forwarding frames while avoid becoming isolated. This may be an optimal strategy if the cost of executing crypto-puzzles is lower than that of forwarding frames and the quarantine period is not sufficient to prevent the node from receiving the stream with minimum reliability. Figure 4 shows the reliability experienced by a non-altruistic node adopting this behaviour. Knowing that, while on quarantine, a node is unable to search for another relationship, the figure depicts reliability values for different quarantine times. It can be observed that if the quarantine time is larger than the detection time, then the proportion of frames received by the non-altruistic nodes, drops significantly. This shows that, with the right choice of the quarantine time, nodes do not increase their utility by adopting this strategy.

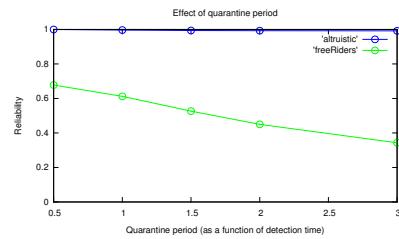


**Figure 3: Recovering from free-riders**

#### 6.5 Operation under Sophisticated Strategies

We now perform a complete analysis of the utility gain of nodes when they may adopt more sophisticated strategies than free-riding and white-washing. Recall that the utility is the difference between the benefits and the communication costs of executing the protocol. Recall also that a strategy can be decomposed into the forwarding, view, and identity strategies. We fix the identity strategy to one identity per node. Later, we discuss the effect of strategies that use multiple identities per node. We analyse the system operation and utility gain when 70% of the nodes remain altruistic whereas the remaining 30% of the nodes follows a deviating strategy. In deviating strategies, nodes may use different values for *baseview* and *bfp* than those specified by FastRank. This covers the strategies with which there may be utility gains: nodes may decrease the *bfp* to decrease the communication costs; they may also increase the size of the active view as a white-washing strategy, with the purpose of circumventing the ranking mechanism, or may decrease it to solve fewer crypto-puzzles.

**System Operation.** We consider a small number of deviating strategies. Regarding the forwarding strategy, nodes can adopt a *free-riding* strategy by not forwarding anything, may follow an *altruistic* strategy by following the protocol, or may follow a *minimum forwarding* strategy, by forwarding the minimum number of frames required to avoid losing relationships. This covers the most relevant possibilities. First, forwarding frames to more nodes on average is equivalent to forwarding frames with the same probability with a higher active view size. Second, forwarding frames with a probability lower than the minimum to keep relationships active is equivalent to free-riding, which as we have seen does not provide any utility gain. Regarding the view, we consider the *enlarged view*, the *shrunk view*, and the *same view* strategies.



**Figure 4: Effect of quarantine period**

Active view size	6	5	4
Altruistic reliability	0.99	0.99	0.99
Non-altruistic reliability	0.88	0.67	0.06
Max hop ratio	1.09	1.37	1.45
Avg hop	1.11	1.13	1.08

**Table 3: Effect of S1**

We evaluate the operation of FastRank under these strategies according to the impact on the reliability of both altruistic and non-altruistic nodes. We show that, whenever non-altruistic nodes may increase their utility, FastRank exhibits the following interesting properties: (i) the reliability experienced by altruistic nodes is only mildly affected, even for large fractions of non-altruistic nodes; (ii) non-altruistic nodes still have to contribute with a reasonable amount of resources to the system in order to get the stream with a minimal reliability; and (iii) if the fraction of non-altruistic nodes is large, then this can be detected by the altruistic nodes.

**S1. Standard bfp with Shrunk View Strategy** This strategy can be beneficial to non-altruistic nodes if keeping fewer relationships still results in a high reliability. In Table 3, we show the reliability of the stream after the deviation and the increase in message latency caused by the deviation, for values of the view size used by non-altruistic nodes smaller than 7. Later, we consider the case where nodes adopt view sizes closer to 12. It can be observed that the reliability of altruistic nodes remains unaffected, whereas the reliability of non-altruistic nodes decreases exponentially for smaller view sizes. This is explained by the fact that non-altruistic nodes consistently end a fraction of their relationships, while altruistic nodes tend to replace those relationships. By configuration, altruistic nodes manage to keep a number of relationships sufficient to keep the network connected among altruistic nodes, thus not impacting their reliability, whereas non-altruistic nodes become disconnected. Another side effect of the considered strategy is that the delay of frame delivery increases significantly for a view size that still provides a significantly high reliability to non-altruistic nodes (15). Consequently, if non-altruistic nodes use a view size sufficiently smaller than 12, either the reliability is too low for them to obtain a utility gain, or the deviation causes a noticeable increase in the delay. In the latter case, an adaptive solution may trigger a change from FastRank to a more costly protocol if the delay of frame delivery surpasses a given threshold, denying a utility gain to non-altruistic nodes in the long term. These nodes may still gain from adopting view sizes closer to 12, but as we show later such strategy does not provide a significant utility gain.

**S2. Minimal Forwarding with Same and Shrunk View Strategies** The advantage of these strategies is that if a node keeps enough relationships, it may still receive the stream with sufficiently high reliability, despite forwarding frames only seldom. Table 4 show the effect of the minimal forwarding with same view strategy. As it can be seen, non-altruistic nodes are severely penalized, as their reliability drops significantly. Even for 30% of non-altruistic nodes, this deviation has no negative impact on the reliability experienced by altruistic nodes. Since the reliability only drops by shrinking the view, non-altruistic nodes clearly do not gain from following the minimal forwarding with shrunk view strategy,

Reliability of altruistic nodes after the deviation	0.97
Reliability of non-altruistic nodes after the deviation	0.50
Fraction of frames sent by non-altruistic nodes	0.2

**Table 4: Effect of S2**

Non-altruists' active view	12	20	25	30	35
Ratio of solved crypto puzzles	1	19	22	42	45
Average frame ratio	0.24	0.40	0.50	0.59	0.66
Reliability of non-altruists	0.71	0.80	0.86	0.98	0.99

**Table 5: Effect of S3 (for one rational node)**

so we opt to omit the evaluation of this strategy.

**S3. Minimum Forwarding with Enlarged View Strategy** The previous results showed that a minimal forwarding strategy is harmful due to a significant decrease in the reliability. A non-altruistic node may circumvent this problem by enlarging the view. Table 4 shows how long it takes for such a node to receive the stream with the same reliability of an altruistic node. It can be observed that a node with a score of *minrank* on all of its incoming links, needs to have a constant active view of size at least 25 to approximate the reliability of an altruistic node. Thus, it need to solve 22 times more crypto-puzzles than altruistic nodes to achieve that state. Furthermore, since it needs to keep all these relationships, it still needs to forward frames, but approximately 80% of that of an altruistic node. Therefore, this strategy is less profitable than altruistic with shrunk view, because with this deviation a non-altruistic node has the same forwarding effort than a node that shrinks its view, with the penalty of performing more crypto-puzzles.

**Utility Analysis.** We now compare the average utility of altruistic nodes with the average utility obtained by a fraction of 30% of non-altruistic nodes. We measure the average utility as the difference between two factors that lie in the interval [0, 1]: (1) the benefits, calculated as the reliability, i.e., fraction of received frames, and (2) the communication costs, calculated according to the following formula:

$$\alpha \times \frac{\text{average number of messages sent by altruists}}{\text{expected number of messages sent by altruists}},$$

where  $\alpha$  is the cost-to-benefit ratio of forwarding and receiving each frame, and the fraction measures the communication overhead relative to a setting with only altruistic node. The factor  $\alpha$  is used to represent two types of utilities, namely, utilities with high ( $\alpha = 1$ ) and low ( $\alpha = 1/2$ ) cost-to-benefit ratios. Given that the expected utility with a reliability close to 1 is roughly  $1 - \alpha$ , and since the expected utility must be strictly positive in order for nodes to be interested in participating in the stream, measuring the utility with  $\alpha = 1$  provides an analysis of the worst-case scenario.

With our initial configuration, we measured an expected number of messages sent by altruists that correspond to an average forwarding probability of 0.18, that is, in a setting with only altruists where nodes use a *bfp* equal to 0.4, the average forwarding probability is 0.18, which is lower than *bfp* due to our ranking mechanism. In our simulations, non-altruistic nodes forward every frame with a fixed probability equal to *bfp* and set *baseview* to *maxview*.

Figure 5 depicts the utilities of altruistic and non-altruistic nodes as a function of the value of *bfp* of non-altruistic nodes,

for active view sizes equal to 8 and 16, and for both types of utilities with high and low cost-to-benefit ratios, respectively. We also measured the average utility for higher active view sizes, but the results are identical to an active view size of 16, so we opt to omit them here. We observe that non-altruistic nodes only obtain a higher utility than altruistic nodes when using an active view size equal to 8, showing that adjusting the quarantine period is effective at denying any utility gain to nodes that increase the active view size, but has the negative consequence of allowing a utility gain to nodes that decrease the active view size. This utility gain is maximal for a forwarding probability of 0.3. The absolute gain is less than 0.2 for high communication costs, and less than 0.04 for low communication costs. Therefore, the utility gain is small relative to the total utility<sup>1</sup>, and decreases for lower values of the cost-to-benefit ratio.

In all cases, we can observe two different trends in the average utility of altruistic and non-altruistic nodes, respectively. Regarding altruistic nodes, the average utility tends to decrease to a minimum around the forwarding probability 0.5, and then it stabilizes roughly constant. This is explained by the fact that, if non-altruistic nodes free-ride, then FastRank quickly isolates them and keeps the reliability of altruistic nodes high. As the forwarding probability increases, altruistic nodes have to forward a higher number of messages to non-altruistic nodes in their views, so their utility decreases despite the increase in the reliability. When the forwarding probability of non-altruists is too high, their average rank increases, obtaining a higher reliability. This causes an inversion of roles, where altruistic nodes forward much less frames than non-altruistic nodes and are expelled from their views. We observe that there is a decrease in both the reliability and communication costs of altruistic nodes at the same rate, such that the average utility remains roughly constant. We note that this negative impact on the reliability of altruistic nodes only occurs for strategies that do not provide a utility gain to non-altruistic nodes.

## 6.6 Heterogeneous Nodes

We repeated the previous experiments in the PlanetLab testbed with 16 nodes and the dissemination of 3000 frames at a rate of 100 kbps. We obtained an optimal configuration where *baseview* is 5, *maxview* is 7, *minRank* is -60, *bfp* is 0.9, and the quarantine period is 16 seconds. Compared to the simulations, the most notable difference is the lower value of *minrank*, which requires a longer quarantine period. This is explained by the greater ranking oscillation between nodes with different upload capacities. Despite this, there is no significant utility variation between altruistic nodes. Specifically, we measured the worst, average, and best utilities of both altruistic and non-altruistic nodes, using the same formula for the average utility as in the previous section with  $\alpha = 1/2$ . The results as depicted in Figure 6 show that the variation in the utility is higher among non-altruistic nodes. This suggests that the dissemination mechanism of FastRank is effective at tolerating some degree of heterogeneity.

## 6.7 Overhead

We establish a comparison between FastRank and related work in terms of the overhead of communication and com-

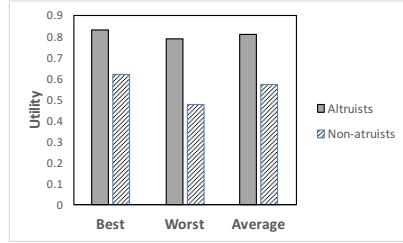


Figure 6: Utility variation in the PlanetLab.

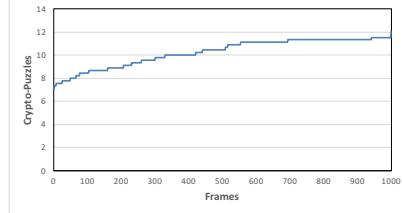


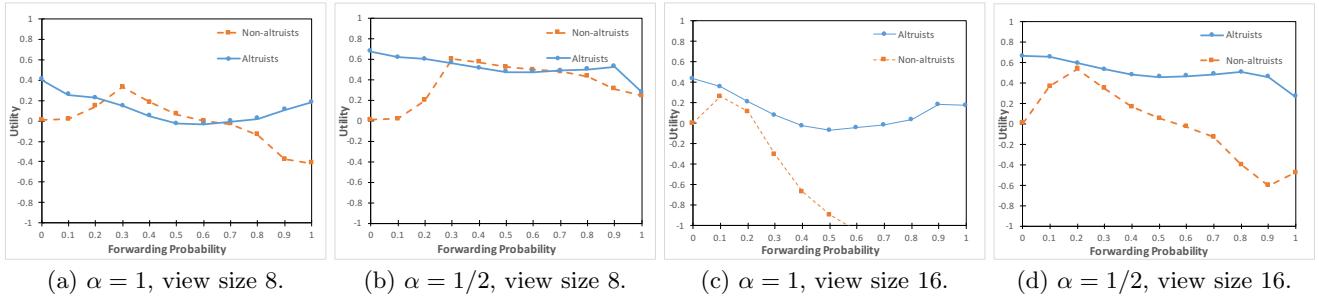
Figure 7: Crypto-puzzles solved by altruistic nodes.

putation. The only non-negligible overhead of FastRank lies in solving crypto-puzzles, which is only performed at the beginning of a relationship. As our results show, we can configure the quarantine period to be sufficiently large to ensure that altruistic nodes only have to solve a number of crypto-puzzles close to the minimum required to build its active view. Due to the presence of free-riders and false-positives in the ranking system, the number of solved crypto-puzzles tends to grow with time during a streaming session. We measured this growth during the dissemination of 1000 frames, in the PlanetLab testbed, where 30% of the nodes are free-riders. Figure 7 depicts the growth of the number of crypto-puzzles solved by altruistic nodes. It can be observed that, initially, the number of solved crypto-puzzles increases quickly, which is due to nodes detecting free-riders and repairing their view, until non-altruistic nodes become isolated and altruistic nodes form a stable sub-network. Clearly, the overhead of solving crypto-puzzles grows sublinearly on the size of the stream. Moreover, in FastRank, nodes never exchange data not related to streaming. This is in contrast with related work, where nodes must periodically exchange additional data for monitoring each others' behaviours. Specifically, in BAR Gossip [15] and FlightPath [14], nodes must sign data and exchange additional monitoring information with a constant size. In LiFTinG [7], no significant computational overhead is imposed, but, since this system uses AV-MON [18], a structured overlay for monitoring nodes, nodes must exchange additional information to maintain the overlay. Moreover, nodes must incur the overhead of performing  $O(f^2)$  direct verifications of the behaviour of other nodes, with a constant probability, where  $f$  is the fanout. In FullReview [4], nodes must log each transmission of a frame, which requires asymmetric cryptography. Moreover, for each node  $i$ , a set of  $M$  monitors of  $i$  must periodically request the log of  $i$  for verification. From these observations, we can conclude that FastRank has the lowest asymptotic overhead.

## 7. CONCLUSIONS

In this paper we have presented FastRank, a peer-to-peer

<sup>1</sup>This suggests that our strategy may be an approximate equilibrium.



**Figure 5: Average utility under sophisticated non-altruistic strategies.**

streaming protocol that relies on an overlay network with symmetric links to mitigate the effect of free riders and white-washers in an efficient and effective manner. FastRank includes overlay construction mechanisms that encourage nodes to perform repeated interactions with a small number of nodes and then leverages from this property to implement efficient localised scoring mechanisms that can be used to deny utility gains to non-altruistic nodes that deviate from the protocol. The resulting system can tolerate up to 30% of non-altruistic nodes without decreasing the reliability of the stream in altruistic nodes. As a result, it allows for an optimised operation in the case where nodes that follow more sophisticated strategies are residual (which happens often in practice). FastRank is in contrast with more robust solutions, that tolerate wider range of deviations at a much higher cost. Interestingly, FastRank limits the utility gain of nodes that adopt more sophisticated behaviours, with small but detectable impact on the perceived streaming process. This open the door for adaptive solutions, where FastRank is used while the number of non-altruistic nodes is small, and the operation reverts to more expensive solutions if the number of non-altruistic nodes adopting more sophisticated behaviours grows to a point where it threatens the reliability of streaming.

**Acknowledgements:** This work has been partially supported by Fundação para a Ciência e Tecnologia (FCT) through projects with references PTDC/EEI-SCR/2776/2012 (PEPITA), PTDC/EEI-SCR/1741/2014 (Abyss) and UID/CEC/50021/2013.

## 8. REFERENCES

- [1] E. Adar and B. Huberman. Free riding on gnutella. *First Monday*, 5(10), Oct. 2000.
- [2] N. Borisov. Computational puzzles as sybil defenses. In *IEEE P2P 2006*, 2006.
- [3] B. Cohen. Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer Systems*, 2003.
- [4] A. Diarra, S. B. Mokhtar, P. L. Aublin, and V. Quéma. Fullreview: Practical accountability in presence of selfish nodes. In *IEEE SRDS*, 2014.
- [5] W.-C. Feng, E. Kaiser, and A. Luu. Design and implementation of network puzzles. In *IEEE INFOCOM 2005*, 2005.
- [6] A. Ganesh, A.-M. Kermarrec, and L. Massouli. SCAMP: peer-to-peer lightweight membership service for large-scale group communication. In *NGC*. Springer-Verlag, 2001.
- [7] R. Guerraoui, K. Huguenin, A.-M. Kermarrec, M. Monod, and S. Prusty. LiFTinG: lightweight freerider-tracking in gossip. In *Middleware*, 2010.
- [8] C. Hauert, M. Holmes, and M. Doebeli. Evolutionary games and population dynamics: maintenance of cooperation in public goods games. *Proc. of the Royal Society of London B: Biological Sciences*, 273(1600):2565–2571, 2006.
- [9] Y. Huang, T. Fu, D.-M. Chiu, J. Lui, and C. Huang. Challenges, design and analysis of a large-scale p2p-vod system. *SIGCOMM Comput. Commun. Rev.*, 38(4):375–388, 2008.
- [10] D. Hughes, G. Coulson, and J. Walkerline. Free riding on gnutella revisited: The bell tolls? *IEEE Distributed Systems Online*, 6(6):1–, June 2005.
- [11] M. Karakaya, I. Korpeoglu, and O. Ulusoy. Free riding in peer-to-peer networks. *IEEE Internet Computing*, 13(2):92–98, 2009.
- [12] A.-M. Kermarrec, L. Massoulié, and A. J. Ganesh. Probabilistic reliable dissemination in large-scale systems. *IEEE TPDS*, 14(3):248–258, 2003.
- [13] J. Leitão, J. Pereira, and L. Rodrigues. HyParView: a membership protocol for reliable gossip-based broadcast. In *IEEE DSN*, 2007.
- [14] H. Li, A. Clement, M. Marchetti, M. Kapritos, L. Robison, L. Alvisi, and M. Dahlin. Flightpath: Obedience vs. choice in cooperative services. In *OSDI*, 2008.
- [15] H. Li, A. Clement, E. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin. Bar gossip. In *OSDI*, 2006.
- [16] X. Liao, H. Jin, Y. Liu, L. Ni, and D. Deng. Anysee: Peer-to-peer live streaming. In *IEEE INFOCOM*, 2006.
- [17] A. Montresor and M. Jelasity. Peersim: A scalable p2p simulator. In *IEEE P2P*, 2009.
- [18] R. Morales and I. Gupta. Avmon: Optimal and scalable discovery of consistent availability monitoring overlays for distributed systems. *IEEE TPDS*, 20(4):446–459, Apr. 2009.
- [19] J. Oliveira, I. Cunha, E. Miguel, M. Rocha, A. Vieira, and S. Campos. Can peer-to-peer live streaming systems coexist with free riders? In *IEEE P2P*, 2013.
- [20] J. Smith. *Evolution and the Theory of Games*. Cambridge University Press, New York, 1982.
- [21] X. Wang and M. Reiter. Defending against denial-of-service attacks with puzzle auctions. In *IEEE Security and Privacy*, 2003.

# Dynamic Pricing for Maximizing Cloud Revenue: A Column Generation Approach

Fadi Alzhouri  
 Department of Electrical and Computer Engineering  
 Concordia University  
 Montreal, Canada  
 f\_alzh@encs.concordia.ca

Anjali Agarwal  
 Department of Electrical and Computer Engineering  
 Concordia University  
 Montreal, Canada  
 aagarwal@ece.concordia.ca

Yan Liu  
 Department of Electrical and Computer Engineering  
 Concordia University  
 Montreal, Canada  
 yan.liu@concordia.ca

Ahmed Saleh Bataineh  
 Concordia Institute for Information Systems Engineering  
 Concordia University  
 Montreal, Canada  
 Ahmedbataineh87@gmail.com

## ABSTRACT

Virtualization underpins the emergence of Infrastructure as a Service (IaaS) to provide service delivery. Even with virtualized resources, the cloud industry suffers from under-utilization off the peak load periods. Hence it is essential for a pricing mechanism to satisfy both customers' demands and budgets as well as maximizing the cloud revenue. The limited works on dynamic pricing consider selling the entire population of VMs in the cloud, resulting in increased processing time and risk. Motivated by the aforementioned concerns, this paper proposes a dynamic pricing model that strives to minimize the inevitable stagnant VMs, while working to maximize cloud revenue. In particular, this paper explores the factors that drive price dynamism following the pay-as-you-go pricing scheme that is rapidly adopted in cloud services. We present a new dynamic pricing approach based on a linear programming relaxation of Stochastic Markov Decision processes. To further reduce the high dimensionality, we employ column generation. Numerical evaluations show practical scalability, as well as prominent influences that could be adopted by cloud providers for revenue maximization.

## CCS Concepts

- Networks → Cloud computing; Network management;
- Mathematics of computing → Discrete optimization;
- Applied computing → Decision analysis; Marketing;

## Keywords

Cloud computing, Dynamic price, Dynamic programming, IaaS, Spot instances, Virtual machines, Column generation

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ICDCN '17, January 04-07, 2017, Hyderabad, India

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007756>

## 1. INTRODUCTION

The IaaS cloud model provides virtualized computing resources as a cost-saving commodity to customers (enterprises, agencies, and individual users) through eliminating the capital expense of the hosting infrastructure. The recent successes in delivering on-demand, scalable, and elastic computing services, and eliminating IT restrictions, underlie the boom that cloud computing has experienced in information technologies. IaaS attracts a variety of enterprises of all sizes and drives a high percentage of enterprise IT budgets. For instance, the 2015 Global Venture Capital Confidence Survey revealed that investing in cloud computing has earned leading positions in the technology market [1]. Gartner, a well-known firm in technology industry research, anticipates that the highest growth in cloud computing market will be for IaaS services up to \$22.4 billion in 2016 [8]. Furthermore, Synergy reports that investments in IaaS cloud service earned \$20 billion in revenues for the year 2015 and witnessed a market growth of 28% [2]. Despite these technical and economic incentives, vast critical issues exist in this fresh technology industry from the perspective of the cloud providers. Maximizing cloud revenue is one of the most critical challenges faced by cloud providers, especially when taking utilization levels into account. Amazon disclosed that utilization levels in 2015 were often under 20% [7]. In general, the massive efforts toward addressing this problem can be categorized as i) minimizing operational costs (e.g. power consumption and cooling systems) [15, 31, 18], and ii) maximizing cloud revenue by creating more profitable pricing models [25, 32, 33, 4]. While price has a significant impact on customers' behavior as it directly affects their budgets, it remains a major concern for cloud providers, as it drives the growth of their profits by allowing them to maximize IaaS cloud revenue. IaaS cloud providers are stimulated to adopt a dynamic pricing strategy. For instance, Amazon launched the first dynamic pricing model in the cloud, Spot instances, to sell its spare computing capacity to customers who are willing to bid a price that at least meets the current spot price [5]. This stimulated many IaaS cloud providers to adopt a dynamic pricing strategy, and encouraged researchers to investigate this trend in cloud as well.

The key role of dynamic pricing is to automatically adjust prices in response to the law of supply and demand, in order to maximize both utilization levels and income. However, most discussions regarding Amazon’s spot pricing demonstrate that it is artificially updated [13, 21, 35]. Spot instances’ pricing history exhibits an unreasonable spike in price for the m2.2xlarge instance in September of 2011 [12, 20]. Furthermore, IaaS cloud providers do not declare much with regard to the adopted mechanism of dynamism. Motivated by the aforementioned issues as well as the success stories of dynamic pricing that have been achieved in widespread applications (e.g. airline and hotel reservations) [29], this paper seeks to answer the following questions:

- Why involve dynamic pricing in cloud computing?
- What type of stochastic process is involved?
- Which factors affect the dynamic pricing of cloud computing?
- How can demands be mapped to stagnant VMs?
- Is it possible to find the optimal policy that promotes the highest revenue in the right amount of time?

The requirements to respond to these questions underline the insights and hence methods, that may be effective in mitigating the problem. Using the general framework provided by the aforementioned questions, this paper contributes the following:

- It proposes a systematic study of the objectives and necessities that underlie dynamic pricing in cloud.
- It characterizes price dynamism parameters and leveraging power consumption as a workload metric.
- It formulates a column generation (CG) approach to linear relaxation of the maximum expected revenue, under discrete-time stochastic Markov decision processes over a finite horizon.
- It validates the proposed approach using numerical results and a case study.

The rest of the paper is organized as follows: In Section 2, we propose our model and formulate the maximum expected revenue under a discrete finite horizon Markovian decision. Section 3 augments column generation for linear relaxation of the model. Evaluation results are realized in Section 4. In Section 5, we review the state of the art work in dynamic VM pricing. We summarize and conclude this paper in Section 6.

## 2. PROPOSED PRICING MODEL

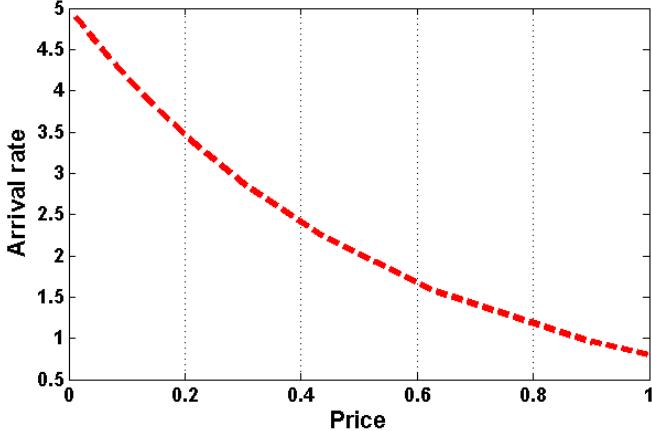
### 2.1 Dynamic price: Objectives and necessities

Answering the first question uncovers the necessities and objectives that drive and encourage cloud providers. The adoption of a dynamic pricing strategy in the cloud by any organization emphasizes its necessity. For instance, Amazon states that the customer bids the maximum price they are willing to pay per instance per hour on Amazon’s spare EC2 instances, and the request is accepted if the bid price exceeds the current Spot price [5]. The Spot price, assigned

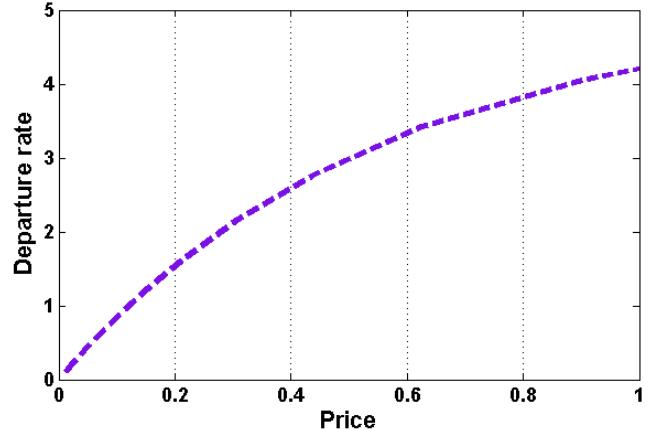
by Amazon, fluctuates such that the price decreases to run the service, or it increases to terminate the service when the demand for the instance increases, or when the instance’s supply decreases. On the other hand, Google presents a finite set of virtual machines for sale in its Preemptible VM Instance, which is only available during non-peak periods and terminates the running instances when the underlying resources are needed [10]. Furthermore, Google deems that the price roughly follows Moore’s Law [11]. It is worth noting that both charge the customer a fixed price per hour during one round, from the beginning until termination. It is evident that they offer a set of VMs for a limited time and at a volatile price, depending on certain conditions, despite seemingly providing a cost-saving service. The question remains as to how these VMs are generated and how they are terminated. Changes in price are an implicit consequence of demand for VMs in reserved and on-demand pricing models. From a technical point of view, virtualizing a physical machine (PM) leads to a limited set of VMs that vary in type and quantity depending on the supply (i.e. the type of available PM), in addition to the demand (i.e. the requested VMs). For example, assume that there is an on-demand or reserved request with targets of type A VM. The cloud accepts this request by operating a PM, which could be scaled up to four VMs of the desired type A. In reality, the cloud loses the potential profit of such machines (taking into account the power consumption and other operational costs), as well as the VMs that were prepared for the sake of high availability and disaster recovery. We argue that a dynamic pricing model, without a service level agreement (SLA), could be the best pricing model for utilizing such virtual machines (called stagnant VMs). For the customer’s perspective, it could attract customers whose jobs have batch or interrupt-tolerant features if it meets their demands (i.e. type, quantity, and price). It is clear that utilizing redundant and randomized sets of stagnant VMs in order to maximize cloud marginal revenue are not trivial objectives, considering the randomness involved in many aspects.

### 2.2 Dynamic price: 3D Stochastic property

Initial analysis of tackling such issues addresses stiff challenges, specifically the three-dimensional (3D) stochastic dilemma. First, uncertainty in arrival rate or purchase demand has been noted in other domains, such as airline seats and hotel booking. Second, efforts made to resolve this problem add yet another dimension of uncertainty, namely the departure rate or demand to leave. Third, the diversity of cloud states increases when there are many VMs, as the cloud fluctuates between active and stagnant states. In fact, the 3D triple stochastic axis dramatically increases the size of the search space for finding the optimal solution. This provides interesting incentives and complex analytical insights. It is worth noting that many researchers in an analogous research domain have utilized the Poisson probability distribution, since it is the most convenient distribution to describe the uncertainty of arrival and departure processes. Furthermore, one of the critical issues that emerges from applying the dynamic pricing approach in an IaaS cloud environment is response time. This imposes an immediate decision, which disappears in other applications where the provider has enough time to accept or reject the request. The following sections outline our approach to overcoming

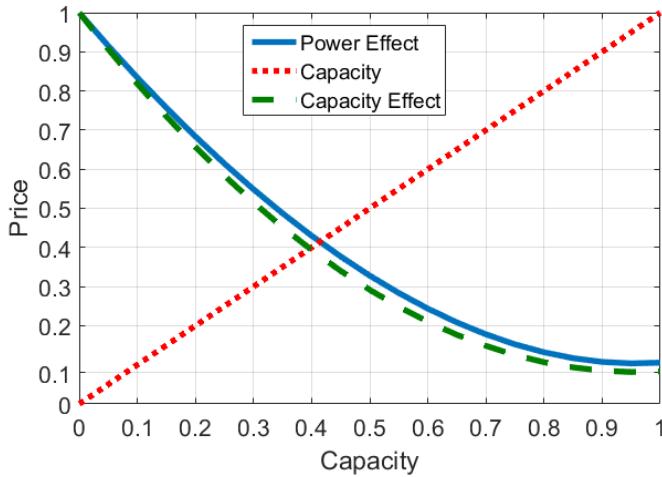


(a) Arrival rate

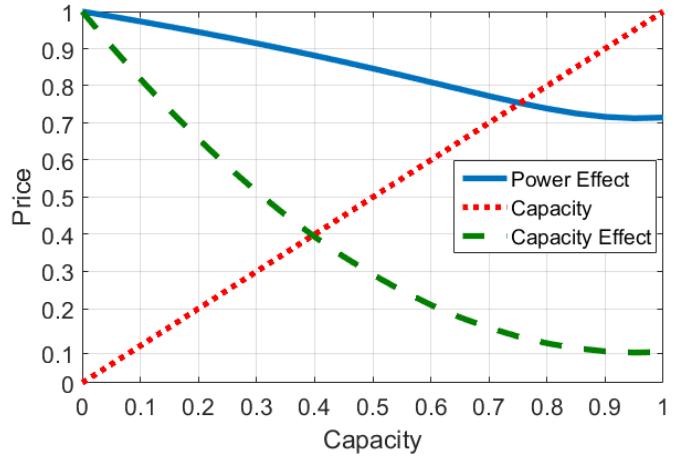


(b) Departure rate

Figure 1: Arrival and departure rates ( $\gamma = 5$  and  $\beta = 5$ )



(a) Low Power Consumption



(b) High Power Consumption

Figure 2: Price vs. Capacity and Power

such obstacles after establishing the dynamism mechanism adopted.

### 2.3 Dynamism parameters

In this section, we strive to define the key factors that have a vital role in pricing dynamism and hence formulate a price in terms of the dominant parameters. To achieve this goal, we capture and quantify the potential implications of parameters employed by various authors [35, 36, 37]. Consequently, we observe that capacity, the number of available VMs, and time (i.e. the moment of sale) are the most prominent parameters. These observations reflect the economic point of view that, on one hand, the quantity of states increases as the price decreases, and vice versa; on the other hand, the more time available to sell, the more flexible opportunity there is to gain [27]. Additionally, one should not lose sight of the actual cost of VMs, which relies on numerous overlapping factors and resources beyond the

scope of this paper. However, one can observe the effect of power consumption costs on total cloud revenue [23] and, more specifically, the correlation between power consumption and virtualization [22, 24]. The latter shows how power consumption is directly proportional to virtualization processes. This insight inspires authors to leverage a periodic power consumption variance as an active player in the dynamic pricing mechanism. Another consequent advantage arises from employing power consumption as an automatic and scalable meter for the purpose of mitigating congestion. To the best of our knowledge, this is the first paper to leverage power consumption in creating a dynamic pricing model aimed at maximizing cloud revenue. To achieve this goal, let  $\rho(t) \in [0, \rho_{max}]$  denote the price of a given VM at time  $t$ , where  $\rho_{max} \leq 1$ . We define the price as a function of power consumption  $w$  and capacity  $c$  at time  $t$  as follows:

$$\rho(t) = ((1 - c(t))^a + \epsilon)^{e^{-w(t)b}} \quad (1)$$

where  $a$ ,  $b$  and  $\epsilon$  are calibration parameters related to the cloud state. Such a definition appears convenient in our case since two discriminating power consumption levels work as sentinel values. Fig. 2 depicts the effect of low power consumption, namely low utilization, and high power consumption, namely high utilization, on the behavior of price in terms of available capacity and time. The former, Fig. 2a, shows that low power consumption implies that the available capacity will become a dominant dynamic parameter in price differentials, where the price  $\rho(t) \simeq (1 - c(t))^a + \epsilon$  when  $wb \simeq 0$ . The empty set of available VMs,  $c(t) = 0$ , allows the price to increase to the maximum value, as an implicit condition that imposes a cloud price of  $\rho(t) = \rho_{max} \simeq 1$  when the stock is zero. In contrast, the cloud forces the price to be zero,  $\rho(t) \simeq 0$ , in the case of redundant capacity  $c(t) \simeq 1$ . The latter, Fig. 2b, reveals how high power consumption significantly affects the price by constricting it to the maximum value as a binding condition, in order to mitigate cloud congestion. This occurs when  $wb \simeq 1$ .

Thus far, our focus has remained on endogenous factors. We now move to discuss the crucial exogenous factors. We adopt the same notions of arrival and departure rate as [35, 14]. We can loosely assume that demands, namely arrival and departure rates, are functions of price at a certain time and that the acceptance or rejection processes do not affect these rates. Furthermore, to model economic perspectives, we assume that:

- $\lambda_t(\rho)$ , the demand to hold a VM of price  $\rho$  at time  $t$ , is non-increasing in  $\rho$  as well as  $\mu_t(\rho)$ , the demand to release a VM of price  $\rho$  at time  $t$ , is non-decreasing in  $\rho$ .
- when price descends to the lowest level  $\rho(t) = 0$ , the demand to hold the VM ascends to the maximum expected value  $\lambda_t(0) = \gamma$ ; as a result, theoretically there is no incentive to leave the system,  $\mu_t(0) = 0$ .
- Conversely, when price ascends to the maximum level  $\rho(t) = 1$ , there is no incentive to hold a new VM  $\lambda_t(1) \simeq 0$ , thus forcing customers to leave cloud services  $\mu_t(1) \simeq \beta$ .
- The mean arrival rate  $\gamma$  and mean departure rate  $\beta$  are non-negative.

Fig. 1 shows an example of demand curves derived from the following formulas

$$\lambda_t(\rho) = \gamma / e^{\alpha\rho(t)} \text{ and} \quad (2)$$

$$\mu_t(\rho) = \beta(1 - 1 / e^{\alpha\rho(t)}), \quad (3)$$

where the scaling parameter  $\alpha \geq 1$ .

## 2.4 Optimum expected revenue

Using this general framework, we create and formulate an infrastructure as a service (IaaS) cloud model that presents and solves the problem, finding the optimum mapping from stochastic demands to available stagnant virtual machines and hence addressing it with the view of maximizing marginal cloud revenue. To fulfill these purposes, Fig. 3 presents a state of an IaaS cloud and assumes that a pool of  $|N|$  stagnant VMs are available for sale under a dynamic pricing

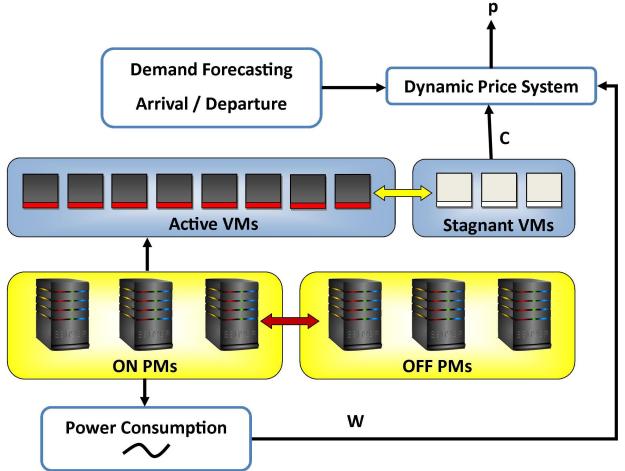


Figure 3: A system model.

model. It is obvious that price discrimination varies based on any change in power, demand, or resources.

The state of the system at time  $t$  is  $C \in \mathbf{C} = \{0, 1, \dots, |N|\}$ , and the expected profit of the current state must be considered within a finite horizon  $H$ , which is divided into  $T$  monitoring epochs  $\Delta t$ , where  $\Delta t = 1$  and  $\Delta T = T$  represent the initial and final epochs, respectively. Poisson processes require that each epoch must be small enough to allow handling only one request. However, the request includes not only one unit of instance, but most likely a set of VMs. Each set  $A \in \{0, 1, \dots, |N|\}$  represents a type of service and directly contributes  $R_A$  fare to the cloud's revenue.

The problem concerning dynamic pricing strategies can be formulated as a finite horizon Markov decision process under uncertainty. We consider a discrete time horizon in which, at any given interval and any possible state of the Markov chain, the system selects the optimal decision, to sell or to wait, by traversing all stochastic possibilities of subsequent rewards until the end of the horizon, with a view to maximizing marginal cloud revenue. A request of type  $A$ , demanded by the customer, launches the cloud market at the beginning of the horizon along with a finite space  $C$  of stagnant VMs supplied by the cloud for sale. The dynamic price system either accepts or rejects the request, according to the results of evaluating the expected profit of demanded resources until closing time, the end of the horizon. Approval transfers the cloud to state  $(C - A, t + 1)$  in the next epoch, raising cloud revenue by  $R_A$ . Refusal maintains the cloud state at  $(C, t + 1)$  without additional revenue. On the other hand, leaving the service changes the space of the stagnant pool by adding  $\bar{A}$  VMs. Thus, if we assume that at any time interval  $(t)$ , the demand to hold the VM occurs with probability  $p(t)$  and the demand to release the VM occurs with probability  $q(t)$ , the system must take the decision  $u_A$  according to the current state  $C$  and the customer's bid  $R_A$ , aiming to maximize the total expected revenue  $f(C, t)$ . This could be evaluated using Hamilton-Jacobi-Bellman's equation, as follows:

$$\begin{aligned}
f(C, t) = \max_{\bar{u} \in U} & \left\{ \sum_{j=1}^n p_j(t)(R_A u_A + f(C - A u_A, t+1)) \right. \\
& + \sum_{j=1}^n q_j(t)f(C + \bar{A}, t+1) \quad (4) \\
& \left. + (1 - (\sum_{j=1}^n p_j(t) + \sum_{j=1}^n q_j(t)))f(C, t+1) \right\},
\end{aligned}$$

subject to the terminal conditions:

$$\begin{aligned}
f_u(C, 0) &\triangleq 0, \\
f_u(C, T+1) &\triangleq 0, \text{ and} \\
f_u(0, t) &\triangleq 0, \quad \forall t, \forall C
\end{aligned}$$

This evaluation can be used because the revenue must be set to zero before launching the market ( $t = 0$ ), after closing the market ( $t = T+1$ ), and when the pool of stagnant VMs becomes empty ( $C = 0$ ). It is worth noting that to fulfill the ultimate objective, the dynamic price system computes  $f^*(C, 1)$ . For more detailed information regarding discrete finite horizon stochastic Markov decision processes, the properties of model formulation (4), and the optimal policy of maximizing revenue, we refer the reader to other research [14, 35].

As mentioned in subsection 2.2, solving (4) in order to find the optimal decision policy  $u^*$  suffers from a triple stochastic space, rendering the problem exhausting and impractical. Consequently, relaxing dynamic programming is a substantial prerequisite for overcoming the imposed high dimensionality.

### 3. COLUMN GENERATION PRICING MODEL

One of the interesting approaches that may be attempted in order to surpass the dimensionality barrier is to approximate dynamic programming based on linear programming (LP). We thus reformulate (4) to maximize the expected objective function  $V$ , by expecting to sell  $X_j$  units of VM using fare  $R_j$  within the horizon as follows:

$$\begin{aligned}
V = \max_X & \sum_j R_j X_j \\
\text{s.t. } & \sum_j a_j X_j \leq C
\end{aligned}$$

$$X_j \leq \sum_t p_j(t) \quad \forall j \quad (5)$$

$$\sum_t q_j(t) \leq X_j \quad \forall j \quad (6)$$

$$X_j \geq 0 \quad \forall j$$

where  $a_j$  represents number of VMs in unit  $X_j$ , and (5) and (6) are arrival and departure constraints, respectively. It is worthwhile that a degree of high complexity arises from the constraints. We employ column generation to address such a large-scale problem and adopt the Dantzig-Wolfe decomposition approach [17].

A column generation approach is especially efficient for solving linear programming problems that sustain large packages

of constraints [26]. Gilmore and Gomory proved this efficiency through their successful practical application when solving the cutting stock problem [19]. This technique works quite well with the revised simplex method, which has a sparse and structured constraint matrix, since the problem could be decomposed into two sub-problems, a master LP problem and an auxiliary LP problem (called the pricing problem). The former's constraints matrix involves a huge number of columns. The explicit enumeration of all columns reach for the optimal solution, but such a process is considerably expensive. The subtle idea proposed by Dantzig and Wolfe states that only a subset of these columns is required to get the solution, because the associated variables of the vast majority of the columns coincide with zeroes in the optimum solution. The later, the pricing problem, is responsible for generating a new, promising column that improves the objective function of the master LP or determining the optimal solution. The new column is added to the master problem, which produces the dual prices (with non-negative reduced cost coefficient). Then the dual solution is passed to the pricing problem. The iterative interactions between master and pricing problem terminate when no new column is generated with a non-negative reduced cost. It is worth mentioning that column generation could be accelerated when the problem is within distance-K tolerance of the optimal solution by terminating the process before generating the rest of the columns that cover the distance K between the current solution (obviously a near optimal solution) and the optimal solution. To fulfill this, a certain number of iterations or a reduced cost criterion could be used. For more information about column generation and Dantzig-Wolfe decomposition, we refer the reader to [16].

## 4. PERFORMANCE EVALUATION

This section records the empirical evaluation results. We involve a column generation approach to solving the linear relaxation of the dynamic programming model. The purpose of this evaluation is to show how well our spot pricing model compares to other spot pricing models. All our empirical evaluations are performed by Matlab version R2015b on an Intel Core i7 machine at 3.6 GHz with 32 GB RAM.

### 4.1 Numerical Results

We consider a discrete time finite horizon Markovian decision in which the system maximizes revenue by selecting the right decision within a certain time. The number of states, namely the capacity of stagnant VM candidates for sale, and the length of horizon are radical criteria for evaluating runtime (or response time). Therefore, we begin by studying runtime for the column generation pricing model (CGPM) and the standard stochastic dynamic programming model (DP). For this purpose, we assume the time interval to be 1 second and the length of horizon to be 3600 intervals. Fig. 4a depicts runtime versus number of available VMs for sale. We discriminate two scenarios; the maximum capacity allowed for spot market is  $C=100$  and  $C=1000$ . It is clear that the runtimes of DP and CGPM are very close when the state is less than 100, whereas there is a clear superiority for DP when the system state varies between 100 and 400 VMs. However, CGPM beats DP if the current state is selling more than 400 VMS (supplying a huge number of VMS). To see the effect of the length of horizon on the performance, we evaluate the runtime over the course of hori-

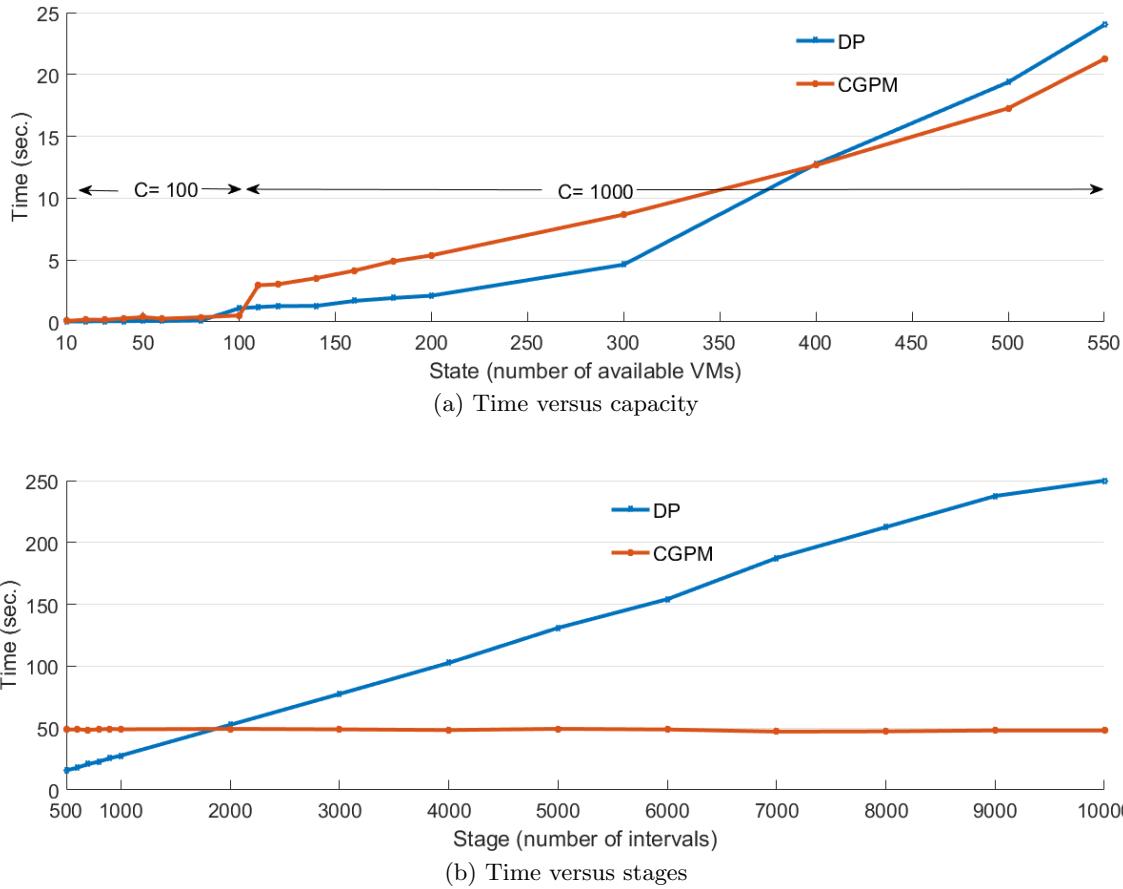


Figure 4: Price versus capacity and stages

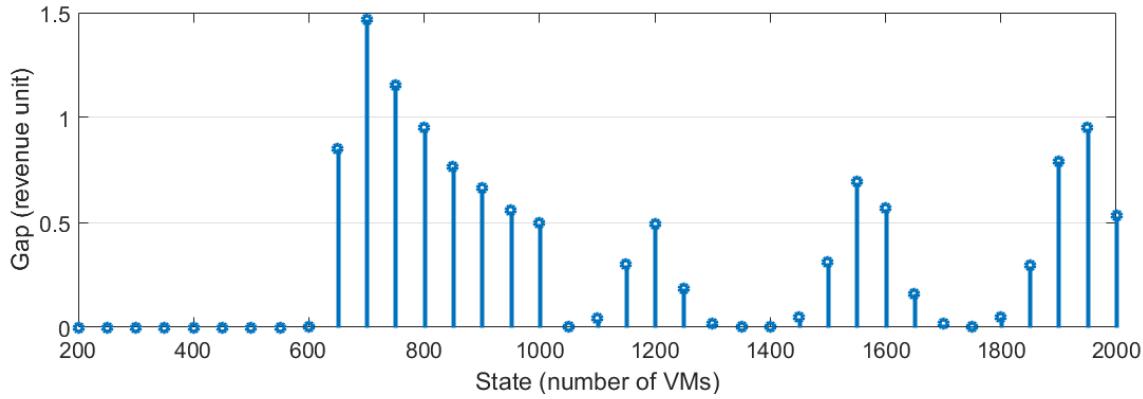


Figure 5: Gap from optimum solution

Case	Profit		Utilization		$\gamma$	$c(t=0)$
	Amazon	CGPM	Amazon	CGPM		
A	12	13.2	4	4	2	3
B	12.4	13.2	5	4	4	3
C	14.5	14.53	4	4	2	3
D	25	25.1	5	5	2	0

Table 1: Profit and Utilization Comparisons

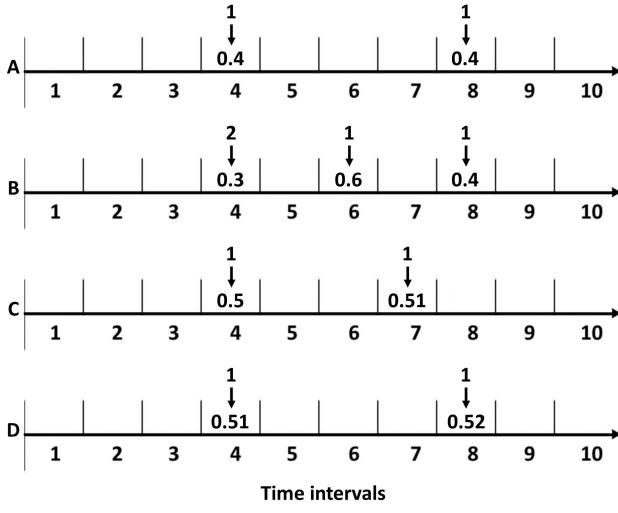


Figure 6: Bids case study: Five VMs available for sale within ten intervals

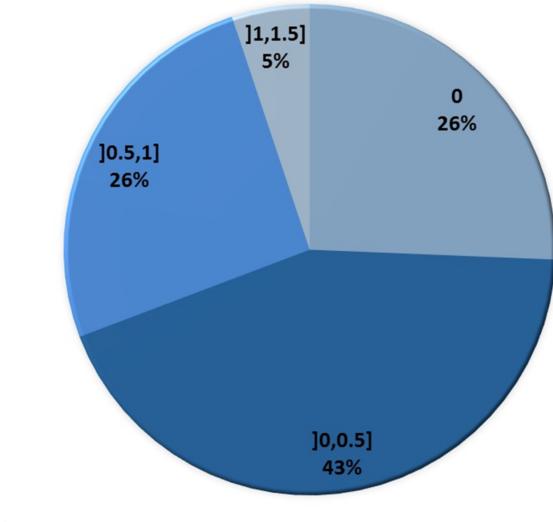


Figure 7: Gap percentage

zons spanning from 500 upto 10000 intervals. Fig. 4b shows that CGPM outperforms DP when the length of horizon exceeds 2000 intervals. The runtime of CGPM shows stability versus the number of stages while DP grows rapidly with it. Moreover, Figs. 5 and 7 depict the gap (error) between the optimal expected solution and the solution obtained by CGPM. Note that in 26% of cases, the solutions are identical (0 gap) and in 43%, the gap is less than 0.5. This indicates that approximately 69% of gaps are less than 0.5. Indeed, this varies depending on the adopted stopping criteria and cutting point used to create the restricted master problem.

## 4.2 Case Study

This case study refers to the pricing strategy of Amazon's Spot instances. To confirm the performance of our model,

we study four cases (A-D), as shown in Fig. 6. In all cases, the horizon is divided into 10 intervals; the number above the vertical arrow represents the number of requested VMs, the number below the vertical arrow is the customer bid for each VM, and the maximum capacity of VMs supplied for spot market  $C = 5$ . Moreover, we assume that at the beginning of the horizon, two VMs have been rented for \$0.50 each. For instance, case A assumes two requests. The first calls for one VM with \$0.40 at time interval 4 while the second also bids \$0.40 for one VM but at interval 8. There are  $c(t=0) = 5 - 2 = 3$  non-utilized VMs at the beginning of the horizon. Table 1 tabulates our findings. The results show that our CGPM model does better than Amazon's strategy [9] in terms of profit, even when utilizing a number of VMs less than Amazon, as in case B. The rationale behind this is that our model rejects the first bid (\$0.30), because it is considered as a worthless bid with high mean arrival rate  $\gamma = 4$ , and then accepts the consecutive bids.

## 5 RELATED WORK

The high financial growth that has been achieved by cloud providers is expected to continue at an intense rate, considering net sales for many organizations [6, 3]. In addition to the frenetic competition between cloud providers to increase cloud utilization, this will lead to a substantial amount of research into creating and addressing a variety of pricing strategies in order to maximize cloud revenue. This review sheds light only on dynamic (or spot) pricing schemes. The story starts with Amazon; their EC2 Spot instances [5] adopt both dynamic price and a pay-as-you-use basis, inspire research, and raise many questions about such hidden strategies as dynamic pricing. As an experiment to reveal the mechanism behind Spot instances, [21] analyzes pricing related to four data centers of EC2 and statistically modeled spot price dynamics as a mixture of Gaussian distribution, but the results were not able to match all types. So does [35] and concludes that the price varies artificially within predetermined price. Moreover, [35] invents a revenue maximization system based on a continuoustime finite horizon, with the stochastic demands of arrival and departure of an IaaS cloud. Based on the latter, [14] innovates relaxed discrete finite horizon Markovian decision processes. Their dynamic pricing scheme and CRM-DP algorithm leverage multi-fare VMs and a considerable runtime. [32] implements a microeconomics approach to find the equilibrium point that satisfies both the consumer's budget and performance, and also the vendor's profits margins. In this, the number of VMs available for each user changes according to the amount of money the user has. However, this is convenient merely in overutilization cases, while [28] employs a genetic algorithm approach in a competitive cloud environment to evolve the price up to a good value, aiming to maximize profits. But the evaluation process assumes that the demand in the morning and evening will be low and high respectively. Also, [30] adopts a price and time slot negotiation (PTN) mechanism, in which the consumer negotiates not only the price, but also time slots with the provider, using a "burst mode" algorithm. Furthermore, auctions induce [34] to model a dynamic auction, where the price changes according to the auctioned capacity and customer's request in terms of number of intervals. Likewise, [25] develops an equilibrium price auction allocation mechanism (EPAAP) and enhances computational time using heuristic and linear programming ap-

proaches. However that auction is economically feasible only for high arrival rates. Later, Amazon reveals some information about the Spot instances' pricing mechanism [9]. On the other hand, Google presents Preemptible VMs with dynamic prices, but only for 24 hours [10]. Among the above, the most related work to this work are [35] and [14]. However, our work excels for cloud spot market with large capacities and long horizons.

## 6. CONCLUSION

This paper proposed a column generation approach to a relaxed dynamic pricing model. The aim is to provide cloud organizations with the ability to utilize spare VMs and hence to maximize cloud revenue. We characterized dynamic pricing necessities, objectives, and parameters. According to those, we formulated the maximum expected revenue using the column generation technique and stochastic Markov processes over a finite horizon. We employed our methodology for studying the performance of cloud spot pricing pioneers. Our proposed approach showed that it is feasible for large-scale Spot instances.

## References

- [1] 2015 Global Venture Capital Confidence Survey. <http://www2.deloitte.com/us/en/pages/technology-media-and-telecommunications/articles/global-venture-capital-confidence-survey.html>. [Online; accessed April-2016].
- [2] 2015 Review Shows \$110 Billion Cloud Market Growing at 28% Annually. <https://www.srgresearch.com/articles/2015-review-shows-110-billion-cloud-market-growing-28-annually>. [Online; accessed April-2016].
- [3] 2016 Top Markets Report Cloud Computing. [http://trade.gov/topmarkets/pdf/Cloud\\_Computing\\_Top\\_Markets\\_Report.pdf](http://trade.gov/topmarkets/pdf/Cloud_Computing_Top_Markets_Report.pdf). [Online; accessed July-2016].
- [4] Amazon EC2 Pricing. <http://aws.amazon.com/ec2/pricing/>. [Online; accessed June-2016].
- [5] Amazon EC2 Spot Instances. <https://aws.amazon.com/ec2/spot/>. [Online; accessed June-2016].
- [6] Amazon.com Announces Second Quarter Sales up 31% to \$30.4 Billion. <https://www.sec.gov/Archives/edgar/data/1018724/000101872416000282/amzn-20160630xex991.htm>. [Online; accessed July-2016].
- [7] Cloud Computing, Server Utilization, and the Environment. <https://aws.amazon.com/blogs/aws/cloud-computing-server-utilization-the-environment/>. [Online; accessed May-2016].
- [8] Gartner Says Worldwide Public Cloud Services Market Is Forecast to Reach \$204 Billion in 2016. <http://www.gartner.com/newsroom/id/3188817>. [Online; accessed June-2016].
- [9] How Spot Instances Work. <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/how-spot-instances-work.html>. [Online; accessed July-2016].
- [10] Preemptible Virtual Machines. <https://cloud.google.com/preemptible-vms/>. [Online; accessed May-2016].
- [11] Pricing Philosophy. <https://cloud.google.com/pricing/philosophy/>. [Online; accessed May-2016].
- [12] Spot Instance Pricing History. <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances-history.html>. [Online; accessed April-2016].
- [13] O. Agmon Ben-Yehuda, M. Ben-Yehuda, S. Assaf, and D. Tsafrir. Deconstructing Amazon EC2 Spot instance pricing. *ACM Transactions on Economics and Computation*, 1(3):1–20, 2013.
- [14] F. Alzhouri and A. Agarwal. Dynamic pricing scheme: Towards cloud revenue maximization. pages 168–173, 2015.
- [15] Q. Chen, P. Grossi, K. van der Veldt, C. de Laat, R. Hofman, and H. Bal. Profiling energy consumption of VMs for green cloud computing. In *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*, pages 768–775, 2011.
- [16] V. Chvatal. *Linear Programming*. Series of books in the mathematical sciences. W. H. Freeman, 1983.
- [17] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.
- [18] J. Galloway, K. Smith, and S. Vrbsky. Power aware load balancing for cloud computing. *Proceedings of the World Congress on Engineering and Computer Science*, I:19–21, 2011.
- [19] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9:849–859, 1961.
- [20] J. HUANG and D. MA. The pricing model of cloud computing services, 2013.
- [21] B. Javadi, R. K. Thulasiram, and R. Buyya. Statistical modeling of Spot instance prices in public cloud environments. In *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, pages 219–228. IEEE, 2011.
- [22] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya. Virtual machine power metering and provisioning. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, pages 39–50. ACM, 2010.
- [23] B. Krishnan, H. Amur, A. Gavrilovska, and K. Schwan. VM power metering: Feasibility and challenges. *SIGMETRICS Perform. Eval. Rev.*, 38(3):56–60, 2011.
- [24] S. Kumar, V. Talwar, V. Kumar, P. Ranganathan, and K. Schwan. vManage: Loosely coupled platform and virtualization management in data centers. In *Proceedings of the 6th International Conference on Autonomic Computing*, pages 127–136. ACM, 2009.

- [25] U. Lampe, M. Siebenhaar, A. Papageorgiou, D. Schuller, and R. Steinmetz. Maximizing cloud provider profit from equilibrium price auctions. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 83–90, 2012.
- [26] M. E. LÃijbbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53:1007–1023, 2005.
- [27] R. Lipsey and C. Harbury. *First Principles of Economics*. Oxford University Press, 1992.
- [28] M. MacÃas and J. Guitart. A genetic model for pricing in cloud computing markets. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, pages 113–118. ACM, 2011.
- [29] B. C. Smith, J. F. Leimkuhler, and R. M. Darrow. Yield management at american airlines. *Interfaces*, 22(1):8–31, 1992.
- [30] S. Son and K. M. Sim. A price-and-time-slot-negotiation mechanism for cloud service reservations. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 42(3):713–728, 2012.
- [31] S. Srikanthiah, A. Kansal, and F. Zhao. Energy aware consolidation for cloud computing. In *Proceedings of the 2008 Conference on Power Aware Computing and Systems*, HotPower’08, pages 10–10. USENIX Association, 2008.
- [32] K. Tsakalozos, H. Kllapi, E. Sitaridi, M. Roussopoulos, D. Paparas, and A. Delis. Flexible use of cloud resources through profit maximization and price discrimination. In *2011 IEEE 27th International Conference on Data Engineering*, pages 75–86, 2011.
- [33] C. Wang, N. Nasiriani, G. Kesidis, B. Urgaonkar, Q. Wang, L. Y. Chen, A. Gupta, and R. Birke. Recouping energy costs from cloud tenants: Tenant demand response aware pricing design. In *Proceedings of the 2015 ACM Sixth International Conference on Future Energy Systems*, pages 141–150. ACM, 2015.
- [34] W. Wang, B. Liang, and B. Li. Revenue maximization with dynamic auctions in IaaS cloud markets. In *Quality of Service (IWQoS), 2013 IEEE/ACM 21st International Symposium on*, pages 1–6. IEEE, 2013.
- [35] H. Xu and B. Li. Dynamic cloud pricing for revenue maximization. *Cloud Computing, IEEE Transactions on*, 1(2):158–171, 2013.
- [36] S. Yi, D. Kondo, and A. Andrzejak. Reducing costs of Spot instances via checkpointing in the Amazon elastic compute cloud. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 236–243. IEEE, 2010.
- [37] L. Z. Zhengping Qian. Distributed systems meet economics: Pricing in the cloud. In *HotCloud ’10*. USENIX, 2010.

# PPRP:Predicted Position Based Routing Protocol Using Kalman Filter for Vehicular Ad-hoc Network

Raj K Jaiswal  
 National Institute of Technology Karnataka  
 Surathkal, India  
 jaiswal.raaj@gmail.com

Jaidhar C D  
 National Institute of Technology Karnataka  
 Surathkal, India  
 jaidharcd@nitk.edu.in

## ABSTRACT

New edition vehicles are equipped with Global Positioning System (GPS) device which provides the vehicle position in the form of latitude and longitude, this position is used as a *location\_id* of the vehicle at time *t* during routing in Vehicular Ad-hoc Network (VANET). The *location\_ids* are susceptible to have an error in position due to several factors such as line-of-sight, signal fading and tunnels just for an instance. Thus, Position based routing protocol experiences poor performance. To minimize the effect of position error, this work proposes a Predicted Position Based Routing Protocol (PPRP) for VANET. PPRP predicts the vehicle location based on previous and current location using Kalman Filter (KF) to improve the Packet Delivery Ratio (PDR), average delay and throughput. Before applying KF into routing its effectiveness is verified and found satisfactory results which advocate KF, to be used in routing. The proposed routing protocol is simulated on NS-3.23 simulator. VANETMOBISIM is used to get the vehicular mobility of 25, 50, 75 and 100 vehicles running on a city road network of  $1000 * 1000 m^2$  area. The performance of the proposed routing protocol is evaluated and compared with other prediction based routing protocol. Simulation is conducted for 250m and 500m transmission range using Winner-II and Two-ray ground propagation model with IEEE 802.11p standard.

## Categories and Subject Descriptors

C.2.2 [Network Protocols]: [Routing Protocols]

## Keywords

Position, Prediction, Kalman Filter, VANET

## 1 Introduction

VANET minimizes traffic, fuel consumption and fatal accidents in road transportation. Along with these services it also enhances the user experience by providing

infotainment services such as nearby gas station, food points etc. in vehicle proximity. It uses IEEE 802.11p communication standard which is designed specifically to be used in VANET to establish vehicle to vehicle and vehicle to roadside unit communication.

VANET is the subset of Mobile Ad-hoc Network (MANET), thus, it has several common features to MANET such as self-configuring node, multi-hop routing, mobility etc. to give some examples. However, in some aspects VANET differs from MANET such as:

- Erratic Communication link: Due to high speed and unpredictable movement of the vehicle, VANET has erratic communication link, causes the end-to-end delay and packet loss.
- Frequent Change in Topology: Due to speed and movement, network topology changes frequently. For instance, a node joins the network and leaves it quickly, incurs the delay, routing overhead and convergence time.
- Hard Delay: In a case of emergency, safety/traffic messages has to reach the destination quickly to ease the traffic or to forbid accident. For instance, information of an accident on the highway should be broadcasted to other incoming vehicles on the same road segment.

Though VANET distinguishes itself from MANET at some aspect, yet MANET routing protocols are used in VANET due to similar characteristics. However, routing performance does not endure with VANET characteristics as MANET routing protocols are designed to work with energy constraint and limited computing power system. Along with these restrictions, it supports for less node mobility and speeds, unlike VANET.

Topology based routing protocols such as ad-hoc on-demand distance vector [16] and optimized link state routing protocol [3] are less suitable in VANET due to node mobility and high speed of the vehicle [8]. Thus, this work focuses only on position based routing.

## Position Based Routing

Position based routing protocol forwards the packets using greedy forwarding. In this protocol, source node computes the Euclidean distance between the neighbors and the destination node and forwards the packet to that node which has the shortest distance to the destination. However,

ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of a national government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only.

*ICDCN '17, January 04-07, 2017, Hyderabad, India*

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007762>

sometimes it happens that the source node forwards the packet to a node which has the shortest distance to the destination, but it is unknown to the source node that the forwarding node is outside of the destination communication range, this is known as a local maximum problem. To mitigate it, different approaches have been proposed such as *perimeter forwarding* in Greedy Perimeter and Stateless Routing (GPSR) [1] and *carry and forward* [10]. Routing protocols such as [4, 12, 18] and [19] took the advantage of confined road structure and navigation system to deploy the anchor nodes to handle local maximum. Though these routing protocols introduced several mechanisms to deal with the local maximum, however, these protocols could not restrict on increasing number of hops, routing overhead and delay either in search of anchor nodes or in *perimeter forwarding*.

Position based routing protocol uses GPS coordinates as a *location\_id* to communicate with other nodes. A node broadcast its *location\_id* at regular intervals to the neighbor nodes and registers it with the location server. A node can retrieve the *location\_id* from location server when destination *location\_id* is not available in *Neighbor\_Table*. The *location\_id* accuracy gets affected by tunnels, buildings, bridge and trees in the city. Apart from these factors, line-of-sight, signal fading, obstruction and interference also affect the accuracy. Thus, accuracy in *location\_id* is a crucial information in VANET communication. Afore-mentioned position based routing protocols did not consider the error in position to improve their performance. According to [17] existing position based routing protocols has following issues:

- Because of the node movement and velocity, location of the vehicle changes frequently. Subsequently, enrolled *location\_id* with location server gets to be outdated within seconds and impacts the routing performance.
- Retrieval of vehicle location using the GPS incurs the delay in routing.
- GPS coordinates get changed unevenly at 100 to 300 meters distance approximately. Consequently, *location\_id* gets to be old to be utilized in routing.
- Accuracy in *location\_id* is also an issue to deliver a packet on time.

In view of afore-discussed issues, this work proposes a position based routing protocol in which location of the vehicle is predicted using KF to improve the PDR and average delay. The notable contribution of this work is:

- It evaluates KF as location prediction in VANET.
- Advance location prediction of the vehicle is also evaluated using KF for highway scenario.
- It proposes position based routing protocol which uses KF for location prediction.
- It measures the routing performance on 250m and 500m of transmission range with Winner-II [7] and Two-ray ground propagation model with IEEE 802.11p standard.
- It compares the performance of the routing protocol with Cross-layer,Weighted, Position-based routing (CLWPR) with prediction.

The remaining sections of this paper are classified as Section 2 discusses the related work, Section 3 describes system model, Section 4 briefs about KF and its result on prediction, Section 5 discusses the proposed routing protocol using KF. Section 6 explains about simulation parameters and simulation results are discussed in Section 7 followed by the conclusion and future work in Section 8. In this paper, node and vehicle, position and location, estimation and prediction are used vise-versa and depict similar meaning.

## 2 Related Work

The following literature uses mobility prediction in routing:

N. Meghanathan [13] proposed a location prediction based routing protocol for MANET. A node broadcast a route request packet when the destination node is not in reach. Location Update Vector (LUV) is added to the route request packet by each node. Based on collective LUV, destination node predicts the topology and sends back a route reply packet. Hamid et al. [14] used movement prediction in GPSR to improve the routing performance. Before sending a packet, the node estimates the next position of the neighboring nodes after  $t$  seconds. A node which is closer to the destination will be selected as the next forwarding node. In their work, prediction mechanism is not discussed. Vinod et al. [15] proposed a prediction based routing for VANET which predicts link expiration time of a route. Their routing algorithm predicts only highway scenario as movement pattern is confined to a particular direction for a long time with a constant speed. Leandro et al. [2] proposed a localization prediction-based routing for VANET. In their work, the node forwards the packet with predicted future location. Prediction mechanism is not discussed in their work. Xue et al. [21] proposed a prediction based routing protocol for VANET which uses variable-order Markov model to predict vehicular mobility pattern.

K. Katsaros et al. [11] proposed the CLWPR protocol in VANET. CLWPR predicts the node position to improve the efficiency of the routing protocol. It predicts the node position using the following equation:

$$Post_t = Post_{t-1} + V_{t-1} * dt \quad (1)$$

In Equation (1)  $Post_t$ ,  $Post_{t-1}$  are the current and previous position, while  $V_{t-1}$  and  $dt$  are previous velocity and time difference.

In the literature review, it is found that work related with routing using mobility prediction is limited.

## 3 System Model

Each vehicle is outfitted with an on-board unit and a GPS device to retrieve its location. The state  $X$  of a vehicle is defined using four-tuple  $[x, y, v_x, v_y]$  where  $x$  and  $y$  are the latitude and longitude position while  $v_x$  and  $v_y$  depicts the speed towards latitude and longitude.

The equation of motion (2), measure the changes in vehicle state. In Equation (2),  $Post_{t-1}$  denotes the initial state of the vehicle and  $V_{t-1}$  and  $a_{t-1}$  depicts the vehicle velocity and acceleration. While  $dt$  represents to the time interval for change in velocity.

$$Post_t = Post_{t-1} + V_{t-1} * dt + \frac{1}{2} a_{t-1} dt^2 \quad (2)$$

Hence, the state model  $X$  of the vehicle is defined as:

$$X = \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} = \begin{bmatrix} x_{t-1} + v_x * dt \\ y_{t-1} + v_y * dt \\ v_{x_{t-1}} \\ v_{y_{t-1}} \end{bmatrix} \quad (3)$$

Here,  $x_{t-1}$  and  $y_{t-1}$  are the state of the vehicle measured at  $t-1$  and  $v_{x_{t-1}}, v_{y_{t-1}}$  are the velocity component.

## 4 KF

KF predict and correct the process state recursively using feedback control mechanism. To obtain the feedback measurement, KF estimates the process state in regular intervals. Equations used in KF are categorized into Prediction and Correction steps.

In Prediction Step,  $\hat{x}_t^-$  is the estimated state of a process using prior state  $\hat{x}_{t-1}^-$  as shown in Equation (4). While  $P_t^-$  is the estimated error covariance on the basis of  $P_{t-1}$  as shown in Equation (5).

In Correction Step,  $\hat{x}_t$  is the estimated system state using observed state  $z_t$ , estimated state  $\hat{x}_t^-$  from Prediction Step and Kalman gain  $K_t$  as depicted in Equation (7). In KF, the accuracy of a state estimation is determined by  $K_t$  which is computed in Equation (6) using  $P_t^-$ , measurement matrix  $H_t$  and measurement noise  $R_t$ . Equation (8) computes the error covariance  $P_t$  using  $P_t^-$  and identity matrix  $I$ .

$\hat{x}_t$  and  $P_t$  from the Correction Step are the final outcomes of the filter as shown in Equations (7) and (8) which are updated with  $\hat{x}_{t-1}^-$  and  $P_{t-1}$  in Prediction Step for next iteration as KF iterates the process to get the best estimation [9] [20].

### Prediction Step

#### Location Prediction

$$\hat{x}_t^- = A \hat{x}_{t-1}^- + B u_{t-1} + w_{t-1} \quad (4)$$

#### Error Covariance

$$P_t^- = A P_{t-1} A^T + Q_t \quad (5)$$

### Correction Step

#### Kalman Gain

$$K_t = P_t^- H_t^T (H_t P_t^- H_t^T + R_t)^{-1} \quad (6)$$

#### Prediction on Measurement $z_t$ (Update)

$$\hat{x}_t = \hat{x}_t^- + K_t (z_t - H_t \hat{x}_t^-) \quad (7)$$

#### Error Covariance (Update)

$$P_t = (I - K_t H_t) P_t^- \quad (8)$$

Remaining symbols used in KF equations are described in Table 1.

The system process model is designed to estimate the system state with the minimal set of information. On the basis of equation of motion (2) and equation (4), process model is designed as:

$$\hat{x}_t^- = A \hat{x}_{t-1}^- \quad (9)$$

The system measurement model provides the system state using measurement device and defined as:

$$z_t = H_t \hat{x}_t^- + e_t \quad (10)$$

Where  $e_t$  is measurement noise at  $t$ .

KF is implemented using the Eigen library for linear

**Table 1: Description of KF Symbols**

Symbol	Description
$A$	State transition matrix.
$A^T$	$A$ transpose.
$B$	Model matrix
$H_t$	(Change in steering and acceleration).
$H_t^T$	Measurement matrix.
$I$	$H_t$ transpose.
$K_t$	Identity matrix.
$P_t^-$	Kalman gain.
$P_{t-1}$	Error covariance.
$P_t$	Prior error covariance at $t-1$ .
$Q_t$	Estimated error covariance by KF.
$R_t$	Process noise covariance.
$u_{t-1}$	Measurement noise.
$w_{t-1}$	Commanded input.
$e_t$	Process noise.
$\hat{x}_t$	Estimated location.
$\hat{x}_{t-1}^-$	Prior location at $t-1$ .
$\hat{x}_t$	KF estimated location.
$z_t$	Measured location.

algebra [5] and C++ programming. In Equation (4),  $w_{t-1}$  and  $u_{t-1}$  are assumed to be zero. To initialize KF, following parameter is set to an initial value.

The transition matrix  $A$  is initialized as:

$$A = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (11)$$

Where  $dt$  is taken as one second.  $H_t$  is defined based on measurement location  $z_t$  which provides the latitude and longitude position of the vehicle. Thus,  $H_t$  is:

$$H_t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (12)$$

The initial state  $X$  is fixed at zero. The error covariance  $P_t^-$  should be large enough to minimize the error gap in state estimation. Thus  $P_t^-$  is:

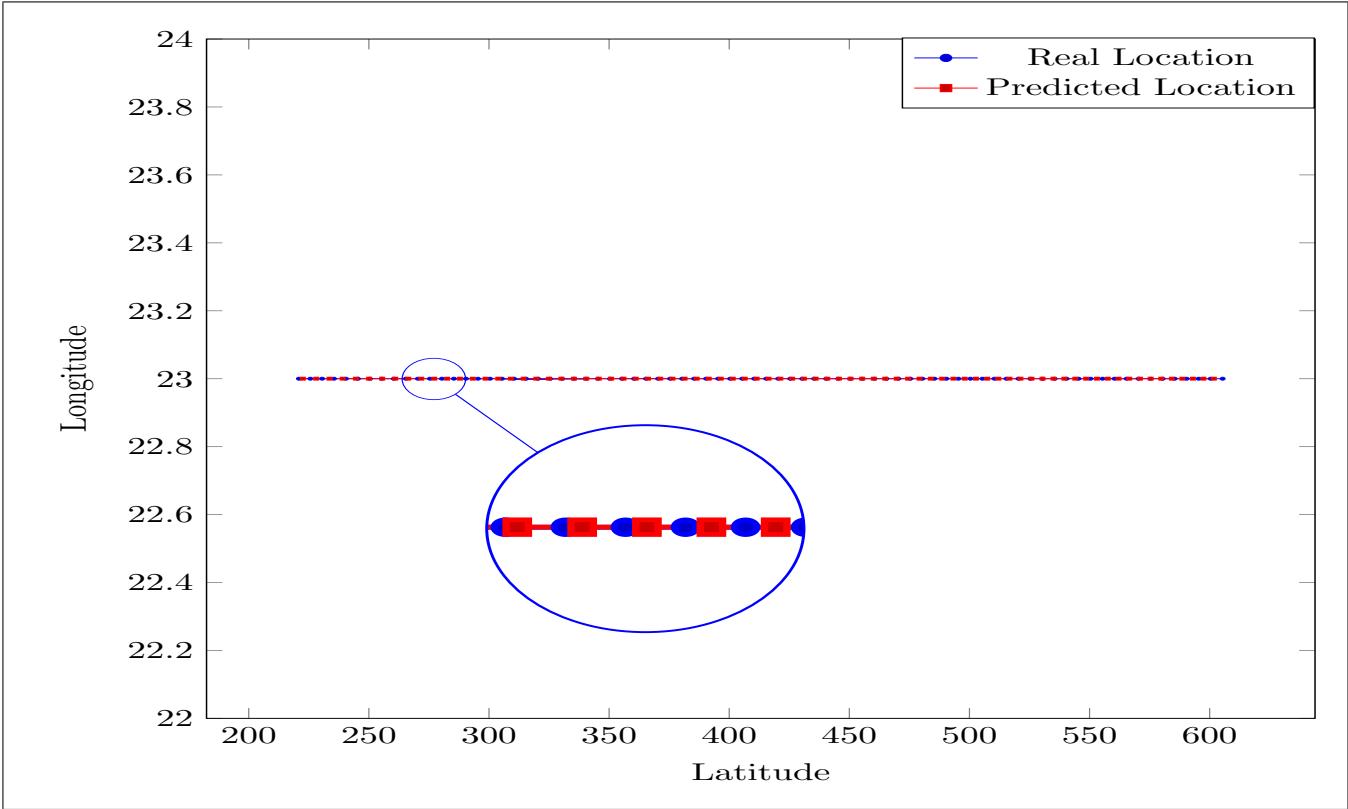
$$P_t^- = \begin{bmatrix} 10000 & 0 & 0 & 0 \\ 0 & 10000 & 0 & 0 \\ 0 & 0 & 10000 & 0 \\ 0 & 0 & 0 & 10000 \end{bmatrix} \quad (13)$$

The process noise  $Q_t$  and measurement noise  $R_t$  can be fixed according to the system model to get the best estimation results. Herein,  $Q_t$  and  $R_t$  are initialized as:

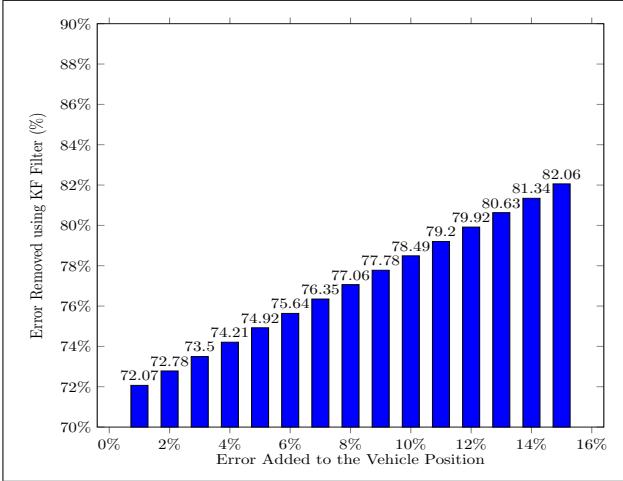
$$Q_t = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix} \quad (14)$$

$$R_t = \begin{bmatrix} 0.001 & 0 \\ 0 & 0.001 \end{bmatrix} \quad (15)$$

Figures 1 and 2 are given just for an instance to support KF. Figure 1 depicts the advance prediction of the vehicle position on the highway. The speed of the vehicle is



**Figure 1: Advance Location Prediction on Highway**



**Figure 2: Error Removal Capacity**

kept constant at 10 m/s. Mobility is generated using VANETMOBISIM [6] in which only latitude position is getting changed while longitude is kept constant to resemble a scenario for a highway.  $Q_t$  and  $R_t$  parameters are tuned to get the best estimation. To get an advance movement prediction of the vehicle on a highway, one-time latitude ( $x$ )= $216.2585823641$  and longitude ( $y$ )= $23.000001$  information supplied to the KF, based on this information KF predicts the advance movement of the vehicle.

Figure 2, shows the error removal capacity of the KF. One time error is added to the  $x$  and  $y$  position. From Figure 2 it is evident that 72.07% of error is removed on the addition of one percent error. The error removing capacity is increased as error increases.

Based on performance instances, KF is applied in routing to estimate the location of the vehicle to improve routing performance as explained in Section 5.

**Table 2: Structure of Neighbor Table**

Node_id	location_id	Velocity
xxxx	x   y   z	xx

## 5 Predicted Position based Routing Protocol

This section discusses the PPRP protocol. PPRP is designed on the basis of position based routing protocol and forwards the packets using greedy forwarding with predicted *location\_id* using KF. Each node maintains the *Neighbor\_Table* which consists the *Node\_id*, *location\_id* and *Velocity* information of the neighboring nodes as depicted in Table 2. Each node advertises its *location\_id* at regular intervals. Subsequently, *Neighbor\_Table* gets updated each time upon a change in *location\_id*.

When a node need to transmit a packet, firstly, it searches the destination *location\_id* in *Neighbor\_Table*, if it is available in the *Neighbor\_Table* then the *location\_id* passed to the KF module to get the predicted *location\_id*. The predicted *location\_id* is added to the destination

**Algorithm 1** Location prediction using KF

---

```

1: procedure KF PREDICTION
2:   Search (Destination location_id in Neighbor_Table
   ↪ at t)
3:   if (Destination location_id is found ) then
4:     Compute (location_id using KF at t+1)
5:     Update (Current location_id ← Predicted
   ↪ location_id)
6:     Send (Packet with predicted location_id to the
   ↪ destination)
7:   else
8:     Send (Request to location server to get
   ↪ destination location_id)
9:     Search (Neighbor node location_id, closer to the
   ↪ destination)
10:    Compute (Neighbor node location_id using KF
   ↪ at t+1)
11:    Update (Current location_id ← Predicted
   ↪ location_id)
12:    Send (Packet to neighbor node with Predicted
   ↪ location_id)
13:   end if
14: end procedure

```

---

address in the packet header. If destination *location\_id* is not available in *Neighbor\_Table* then source node searches the neighbor node which is closer to the destination and passes the neighbor node *location\_id* to KF to get predicted position as shown in Algorithm 1.

## 6 Simulation Parameters

To implement PPRP, CLWPR NS-3.23 patches are used in which KF based prediction module is incorporated. Table 4 outlines the network parameters used in NS-3.23 simulator. VANETMOBISIM is used to generate the vehicular mobility, as it gives more realistic vehicular traffic. Mobility is generated for 25, 50, 75 and 100 vehicles to get the sparse and dense network scenario. All the vehicles are having equal length and deployed in an area of 1000 x 1000 m<sup>2</sup>, which is similar to city road network. Table 3 enlist the details of the simulation parameters used in mobility generation. Simulation is conducted on Two-ray ground and Winner-II propagation model for 250m and 500m transmission range.

## 7 Results and Discussion

PPRP performance is compared with the CLWPR routing protocol as only these two protocols use location predictions. PDR, throughput and average delay are computed on Two-ray ground and Winner-II propagation model for each scenario. PPRP routing overhead is similar to the CLWPR, as PPRP is differed only in location prediction compared to CLWPR. In Figures 3 to 8, TG and WN are used as an acronym for Two-ray ground and Winner-II propagation model.

### 7.1 PDR

PDR is the ratio of a number of the packets received and the number of packets send. Figures 3 and 4 show the PDR for 250m and 500m transmission range. These results are discussed as follows:

**Table 3: Parameters for Mobility Generation**

Description	Values
Simulator	VANETMOBISIM
Simulation Time (s)	499
X (m)	1000
Y (m)	1000
Traffic lights	10
Traffic Light Duration	60s
No. of Lanes	2
Min. Velocity (m/s)	0.1
Max. Velocity (m/s)	25
Driver Politeness Factor	0.8
Max. Acceleration	0.9(m/s <sup>2</sup> )
Max. Deceleration	0.6(m/s <sup>2</sup> )
Min. Congestion Distance	2m
Safe Headway Time	2s
Vehicle Length	4m
Traffic Type	Homogeneous

#### 7.1.1 PDR on 250m Transmission Range

With reference to Winner-II, PPRP-WN has the highest PDR of 56.4% for 25 vehicles and it reduces as the number of vehicle increases, PDR reaches to 25.31% for 100 vehicles. Whereas CLWPR-WN has 54.15% maximum PDR for 25 vehicles and reduced to 20.03% for 100 vehicles as shown in Figure 3. With reference to Two-ray ground model, PPRP-TG has 56% highest PDR for 25 vehicles and reduces as vehicles number increases. Whereas, CLWPR-TG achieve 54.8% maximum PDR as shown in Figure 3. PDR on both the propagation model is better with PPRP using KF compared to CLWPR.

#### 7.1.2 PDR on 500m Transmission Range

PDR of both the protocol on 500m transmission range is better compared to 250m. However, CLWPR-WN has better PDR on Winner-II propagation model compared to PPRP-WN. In sparse networks of 25 vehicles, 82.2% PDR is measured with Winner-II model for CLWPR-WN while it is reduced to 76.2% with PPRP-WN. In both the protocols, PDR has a reciprocal relation to the number of vehicles. It reduced to 40.53% and 38.94% for CLWPR-WN and PPRP-WN when vehicle density increased to 100 as showed in Figure 4. With reference to Two-ray ground model, PDR is less for PPRP-TG compared to CLWPR-TG for the sparse network. However, PPRP-TG performance is better compared to CLWPR-TG when network tends to get denser. 95.7%, 94.9% and 54.4%, 55.4% are the maximum and minimum PDR for CLWPR-TG and PPRP-TG respectively as shown in Figure 4. Overall PDR improves marginally in PPRP protocol with location prediction using KF.

## 7.2 Throughput

Network throughput measures the average number of bytes received per second. Throughput is getting improved in PPRP due to improved PDR. Figures 5 and 6 show the throughput of the protocols for each scenario discussed as follows:

#### 7.2.1 Throughput on 250m Transmission Range

The throughput of both protocols with Winner-II model are 54.74 kbps and 72.12 kbps in sparse networks of 25 vehicles for CLWPR-WN and PPRP-WN. It is reduced as the network gets denser in both the protocols to 25.47 kbps and 31.77 kbps for CLWPR-WN and PPRP-WN in dense networks of 100 vehicles. However, throughput is

**Table 4: Network Simulator Parameters**

Parameters	Values
Network Simulator	NS-3.23
Simulation Time (s)	300
Routing Protocols	CLWPR & PPRP
Antenna Model	Omni-Directional Antenna
Modulation Technique	OFDM
Radio Propagation Model	Winner-II and Two-ray Ground
Transmission Range (m)	250 & 500
MAC Type	IEEE 802.11p
MAC Rate	6 Mbps
Transport Protocol	UDP
Data Type	CBR
CBR Generation Rate(Kbps)	128
Packet Size	500 Bytes
No. of Connections	30% of the Total Number of Vehicles
No. of Vehicles	25,50,75,100

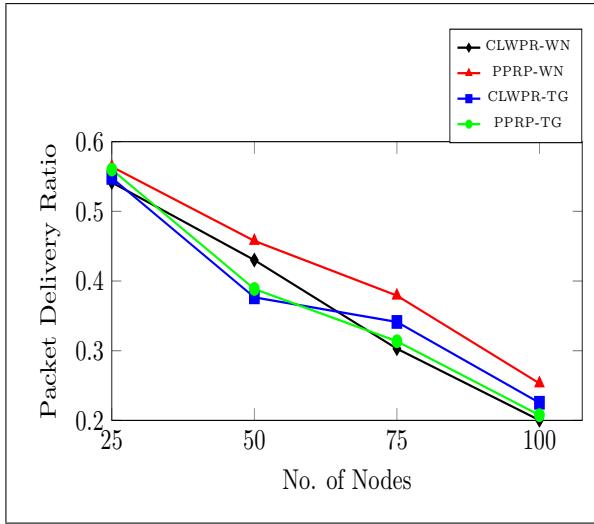


Figure 3: PDR on 250m Transmission Range

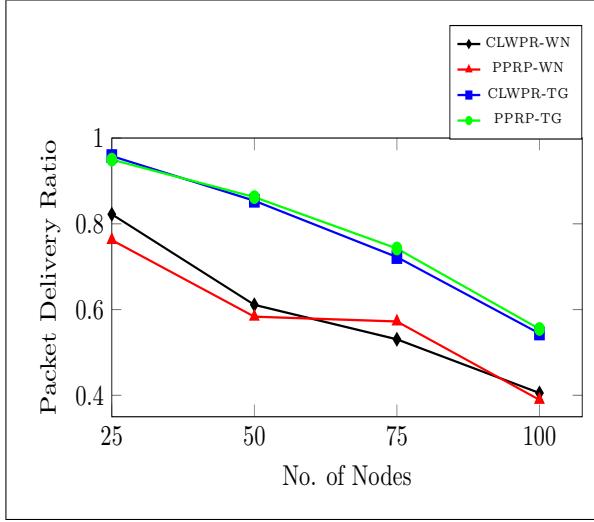


Figure 4: PDR on 500m Transmission Range

quite better in PPRP-WN unlike CLWPR-WN. Similarly, the throughput of PPRP-TG on Two-ray ground model is slightly better than CLWPR-TG in sparse networks.

In the dense network, CLWPR-TG has better throughput compared to PPRP-TG as showed in Figure 5.

### 7.2.2 Throughput on 500m Transmission Range

Throughput gets enhanced in the 500m transmission range for both the protocol on Winner-II and Two-ray ground model. Proposed routing has more throughput compared to CLWPR in all scenarios. 122.49 kbps and 121.43 kbps are the maximum throughputs measured for CLWPR-TG and PPRP-TG in a sparse network and it reduces to 68.49 kbps and 70.23 kbps in dense networks of 100 vehicles using Two-ray ground model. With reference to Winner-II, 97.10 kbps and 105.40 kbps are the maximum measured throughput for CLWPR-WN and PPRP-WN, while it reduced to 47.09 kbps and 50.85 kbps with 100 vehicles network, respectively. Using KF throughput get enhanced with proposed model and it is better in 500m range compared to 250m as shown in Figures 5 and 6.

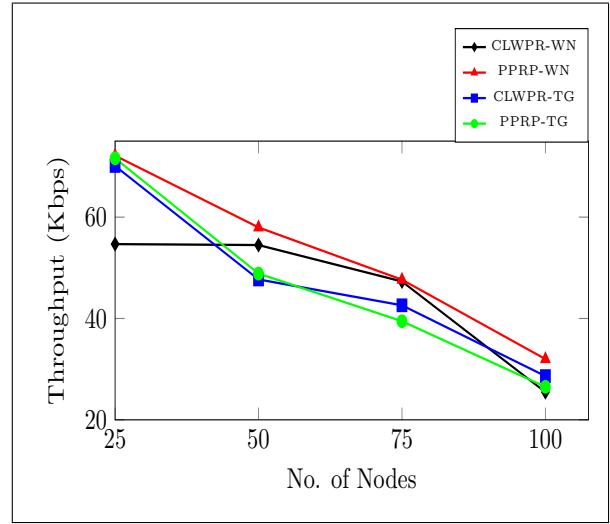


Figure 5: Throughput on 250m Transmission Range

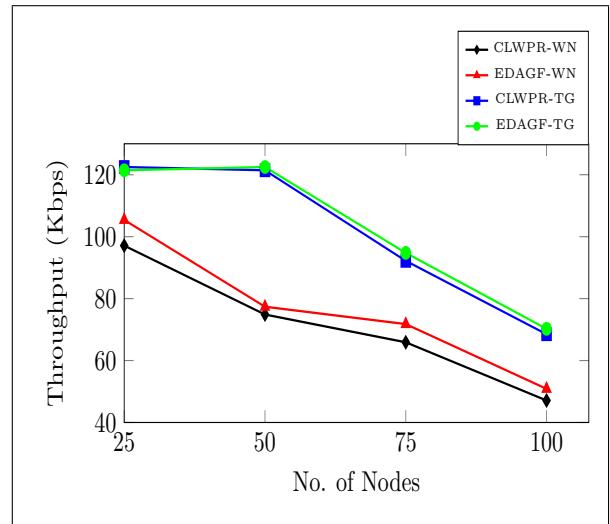


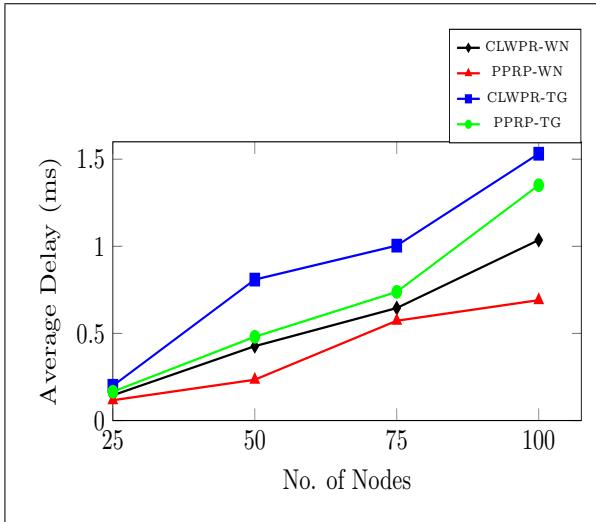
Figure 6: Throughput on 500m Transmission Range

### 7.3 Average Delay

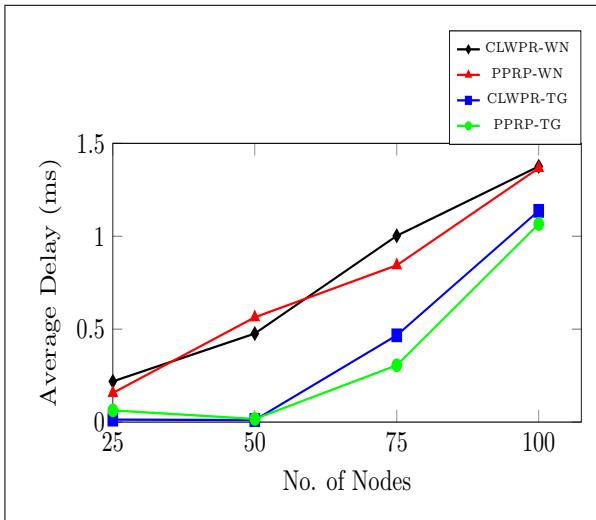
It measures end-to-end transmission delay between source to destination only for correctly received packets. Figures 7 and 8 show the average delay for each scenario as discussed follows:

#### 7.3.1 Average Delay on 250m Transmission Range

PPRP minimizes the delay using KF compared to CLWPR on both the propagation model. Overall, PPRP-WN has a minimum delay among all and it increases as the number of the vehicle increases as shown in Figure 7. Among all 0.116 ms and 0.697 ms are the minimum and maximum delay using PPRP-WN. While, 0.19 ms and 1.53 ms are minimum and maximum delay using CLWPR-TG. Hence, use of KF in 250m range minimizes the delay.



**Figure 7: Average Delay on 250m Transmission Range**



**Figure 8: Average Delay on 500m Transmission Range**

#### 7.3.2 Average Delay on 500m Transmission Range:

With reference to Winner-II, delay is less with PPRP-WN in all vehicle density scenarios except for 50 vehicles. 0.156 ms and 1.36 ms are the minimum and maximum delay observed using PPRP-WN, while for CLWPR-WN it is 0.21 ms and 1.37 ms. With reference to Two-ray ground maximum and minimum delay recorded for PPRP-TG are 1.06 ms and 0.062 ms. Using CLWPR-TG maximum and minimum delay observed are 1.13 ms and 0.01 ms. Delay increases as vehicle number increases. Using KF, it is observed that delay is minimized in all the scenarios.

## 8 Conclusion & Future Work

Based on the results as shown in Figures 1 and 2 it is observed that KF is effective in location prediction and fit to be used in VANET environment. These performances motivated us to use KF into the routing to improve the performance particularly PDR and average delay. Based on results, it is observed that location prediction using KF improves the PDR and average delay compared to CLWPR. It is also observed that PPRP is more effective on Two-ray ground compared to Winner-II propagation model. In addition, PPRP does better on 500m transmission range, unlike 250m. As a future work, the performance of PPRP will be evaluated on heterogeneous traffic, highway traffic and with different size of data packet.

## 9 References

- [1] K. B and K. HT. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proceeding of Mobicom, Conference on Mobile Computing Networking*, pages 243–254. ACM, 2000.
- [2] L. N. Balico, H. A. B. F. Oliveira, R. S. Barreto, A. A. F. Loureiro, and R. W. Pazzi. A prediction-based routing algorithm for vehicular ad hoc networks. In *2015 IEEE Symposium on Computers and Communication (ISCC)*, pages 365–370, July 2015.
- [3] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). RFC 3626, IETF, Oct. 2003.
- [4] F. Giudici and E. Pagani. Spatial and traffic-aware routing (star) for vehicular systems. In L. Yang, O. Rana, B. Di Martino, and J. Dongarra, editors, *High Performance Computing and Communications*, volume 3726 of *Lecture Notes in Computer Science*, pages 77–86. Springer Berlin Heidelberg, 2005.
- [5] G. Guennebaud, B. Jacob, et al. Eigen v3. <http://eigen.tuxfamily.org>, 2010.
- [6] J. Härrä, F. Filali, C. Bonnet, and M. Fiore. Vanetmobisim: Generating realistic mobility patterns for vanets. In *Proceedings of the 3rd International Workshop on Vehicular Ad Hoc Networks, VANET '06*, pages 96–97., New York, NY, USA, 2006. ACM.
- [7] I. IST-WINNER. Deliverable 1.1. 2 v. 1.2, ist-winner2. Technical report, Tech. Rep., 2008 (<http://projects.celti-c-initiative.org/winner+/deliverables.html>), 2007.
- [8] R. Jaiswal and C. Jaidhar. An applicability of aodv and olsr protocols on ieee 802.11p for city road in vanet. In *Internet of Things, Smart Spaces, and Next*

- Generation Networks and Systems*, volume 9247 of *Lecture Notes in Computer Science*, pages 286–298. Springer International Publishing, 2015.
- [9] R. K. Jaiswal and C. D. Jaidhar. Location prediction algorithm for a nonlinear vehicular movement in vanet using extended kalman filter. *Wireless Networks*, pages 1–16, 2016.
  - [10] M. Jerbi, R. Meraihi, S.-M. Senouci, and Y. Ghamri-Doudane. Gytar: Improved greedy traffic aware routing protocol for vehicular ad hoc networks in city environments. In *Proceedings of the 3rd International Workshop on Vehicular Ad Hoc Networks, VANET '06*, pages 88–89, New York, NY, USA, 2006. ACM.
  - [11] K. Katsaros, M. Dianati, R. Tafazolli, and R. Kernchen. Clwpr x2014; a novel cross-layer optimized position based routing protocol for vanets. In *Vehicular Networking Conference (VNC), 2011 IEEE*, pages 139–146, Nov 2011.
  - [12] K. C. lee, J. Haerri, U. Lee, and M. Gerla. Enhanced perimeter routing for geographic forwarding protocols in urban vehicular scenarios. In *IEEE GLOBECOM workshop*, pages 1–10, 2007.
  - [13] N. Meghanathan. Location prediction based routing protocol for mobile ad hoc networks. In *IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference*, pages 1–5, Nov 2008.
  - [14] H. Menouar, M. Lenardi, and F. Filali. Movement prediction-based routing (mopr) concept for position-based routing in vehicular networks. In *2007 IEEE 66th Vehicular Technology Conference*, pages 2101–2105, Sept 2007.
  - [15] V. Namboodiri and L. Gao. Prediction-based routing for vehicular ad hoc networks. *IEEE Transactions on Vehicular Technology*, 56(4):2332–2345, July 2007.
  - [16] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561 (Experimental), July 2003.
  - [17] J. C. D. Raj K Jaiswal. Edagf: Estimation & direction aware greedy forwarding for urban scenario in vehicular ad-hoc network. In *UIC-ATC-ScalCom-CBDCom-IoP, 2015 IEEE*, pages 814–821., 2015.
  - [18] B. C. Seet, G. Liu, B. Lee, C. Foh, K. Wong, and K. Lee. A-star: A mobile ad hoc routing strategy for metropolis vehicular communications. In *Lecture notes in computer science, Networking*, pages 989–999. Springer, 2004.
  - [19] J. Tian, L. Han, and K. Rothermel. Spatially aware packet routing for mobile ad hoc inter-vehicle radio networks. In *Intelligent Transportation Systems, 2003. Proceedings. 2003 IEEE*, volume 2, pages 1546–1551 vol.2, Oct 2003.
  - [20] G. Welch and G. Bishop. An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA., 1995.
  - [21] G. Xue, Y. Luo, J. Yu, and M. Li. A novel vehicular location prediction based on mobility patterns for routing in urban vanet. *EURASIP Journal on Wireless Communications and Networking*, 2012(1):1–14, 2012.

# A Novel Priority Based Exigent Data Diffusion Approach for Urban VANets

Chinmoy Ghorai<sup>\*</sup>

Department of Information Technology  
 Indian Institute of Engineering Science and  
 Technology (Formerly BESUS)  
 Shibpur, Howrah, West Bengal, India 711103  
 chinmoy@it.iests.ac.in

## ABSTRACT

With the rapid development of urbanization, the tendency of road accidents is also highly increasing due to heavy traffic. Most of the victims give up their lives because of the insufficient infrastructure of emergency information distribution to the proper authorities and lack of critical care on urgent basis. Besides restricting the preventable causes like over-speeding, drunk-driving and overloading, it is very important to propagate emergency messages across the network to deal with such disasters. This paper presents a novel scheme for disseminating messages on priority basis over VANets in urban scenarios, which take care of the fairness of the scheduling process by introducing precedence on the emergency messages like the lifesaving message or other critical message over general messages like traffic condition, fuel pump and road condition and or all other commercial messages. For this a laxity based priority scheduling technique with back-off mechanism is introduced. The next packet is to be transmitted based on certain parameter like: its priority considering the laxity budget of the packet, the present packet delivery ratio, required packet delivery ratio and the request selection precedence. To provide the fairness in the scheduling process the request selection precedence is introduced which scheduled the most urgent request for service in the next iteration. The EXata simulator has been extensively used to assess the proposed scheme. The proposed method has been found to work satisfactorily with a number of test cases and considerably outperforms compare to the existing technique.

## CCS Concepts

- Networks → Network protocol design;

---

<sup>\*</sup>Mr. Ghorai is the corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [Permissions@acm.org](mailto:Permissions@acm.org).

ICDCN '17 January 04-07, 2017, Hyderabad, India

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007755>

Indrajit Banerjee

Department of Information Technology  
 Indian Institute of Engineering Science and  
 Technology (Formerly BESUS)  
 Shibpur, Howrah, West Bengal, India 711103  
 ibanerjee@it.iests.ac.in

## Keywords

Vehicular Ad-hoc Network (VANET); Data Diffusion Technique; Packet Delivery Ratio (PDR); Laxity Budget; Intelligent Transportation System (ITS).

## 1. INTRODUCTION

Vehicle communication has presently been an admired research issue. As vehicular ad hoc networks are acquiring popularity day by day a large number of researchers have envisaged in this area. By taking advantages of these networks a large number of vehicles can take right decision by utilizing the information they have. The main basis of VANets is to develop intelligent transportation system (ITS) by providing support for traffic protection, collision caution, advice of lane change, danger caution, and better navigation. IEEE 802.11p is standardized for vehicular communications which are supported on dedicated short-range communications (DSRC) technology which functions at 5.9-GHz band and offers higher data rate up to 27Mbps [1] [2]. In present scenario the main challenges of VANets are high velocity, frequent topology changes and large number of vehicles. In this paper a distributed scheduling scheme has been proposed, where scheduling decisions are made not merely considering the states of neighbour vehicle, as well on feedback from the destination vehicle concerning packet losses. To provide the fairness in this scheduling process the request selection precedence is introduced which scheduled the most urgent request for service in the next iteration.

The rest of the paper is organised as follows. Section 2 overviews the related work and motivation in this field. Section 3 presents brief definition of some measurable parameters. Section 4 presents the proposed data diffusion technique. Section 5 assesses the performance of the proposed technique. Finally, conclusion of this paper is given in Section 6.

## 2. MOTIVATION & RELATED WORK

Having a safe journey by the road is one of the major questions in today's busy life in any congested road in an urban scenario. Due to street traffic caused an approximate 1.25 million deaths worldwide according to the World Health Organization. That is, one person is killed every 25 seconds. Whatever the reason may be, it is most important to distribute emergency information regarding accidents to the proper authorities like nearby traffic control unit, hospi-

tal or fire-brigade, so that a prompt action can be taken on priority basis for managing such disasters and saving lives. For dealing with such issues on time, a distributed laxity based priority scheduling scheme has been carried out so that emergency messages reach the right place on a priority basis by using the Intelligent Transportation Systems.

In recent years, as vehicular ad-hoc networks become popular, researchers have revealed a lot attention into this area. Most of them focus on vehicle routing and other network centric advance. Though, also a number of researchers consider data management as confront. Data scheduling is one of the most key facet in data management. In the past several schemes have been developed by the researchers for packet scheduling like, first-come-first-serve (FCFS), most request first (MRF), longest waiting time (LWT), shortest data first (SDF) and maximum quality increment first (MQIF).

In the [3], Liu et al. proposed a cooperative data scheduling (CDS) to maximize the number of vehicles that recover their requested data. Here, scheduling assessments are made by transforming CDS to maximum weighted independent set (MWIS). In the [4], Cheng et al. proposed an enhanced WAVE MAC by using the time division access on service channels and a channel scheduling scheme has been projected to consult and distribute the channel resources. In the [5], Nakamura et al. proposed a system for competent message delivery in VANETs using the route information in car navigation systems. In this system, each car exchanges the information on its present location at regular intervals and scheduled route in the car navigation system with neighbouring cars within radio range. By referring to the swapped information, each car forwards messages to the neighbouring car that will most closely approach to the destination.

In the [6], Demesure et al. proposed a new navigation approach for mobile agents in Autonomous Guided Vehicles - based flexible manufacturing system. A scheduled motion planner and a priority-based intercession scheme are merged for the navigation method. To solve the conflicts when mediators navigate close to each other or towards the similar resource a priority strategy is introduced using a negotiation practice. Zhang et al. in [7] proposed a D\*S scheduling procedure with the competent search pruning method and further proposed D\*S/N procedure which employ broadcasting to provide several request concurrently. Here, the scheduling procedures are used to schedule the most appropriate request for downloading and uploading at Road Side Unit so that the bandwidth utilization is high and more vehicles get served. In [8], Zhang et al. have introduced vehicle platoon aware data access (VPADA) method in which the data is replicated in the similar platoon to be used by the other vehicles. When the vehicle departs platoon, it pre-fetches necessary data and copied its buffered data to little other vehicle which has recently attached the platoon. V-PADA works to recognize nodes entering as well as exiting particular platoon. Also, it helps to lead to platoon members to duplicate and fetch most appropriate data with high accessibility with least overhead.

Yen et al. in [9] introduced a flooding-limited and multi-constrained QoS multicast routing technique which reduces

route discovery time using minimum resources. It decreases the probability of selection of unwanted nodes and attempts to optimize the routes. Li et al. in [10] proposed a multi-cast protocol named codepipe, which offered opportunity to develop multicast throughput using inter-batch and intra-batch coding method. The proposed proposal efficiently progresses throughput, energy efficiency and fairness with the help of opportunistic routing and random linear coding. Meng et al. in [11] proposed and analysed a spatial reusability-aware single path routing as well as any path routing and progresses throughput. In the [12], Zeng et al. have introduced routing protocol for delay tolerant networks (DTNs). This scheme is unlike from other existing DTN based protocols in the sense that it takes care of energy as well. Author by using Nash Q-learning approach presents a directional routing and scheduling scheme (DRSS) for green vehicle DTNs. In the [13], Zhou et al. converse about heterogeneous media provisioning in peer-to-peer based VANETs and expands fully dynamic system which offers maximum user fulfilment as well as positive amount of contentment in fairness by designing a user satisfaction model. Beside this a joint content dissemination and cache update system is proposed by the author.

### 3. DEFINITION OF SOME MEASURABLE PARAMETERS

Here, a packet switched multi-hop wireless network has been considered, where vehicles can move randomly with varying speed. A unique identifier (id) is applied to each vehicle. All the vehicles which are in direct transmission range of the sender vehicle can receive the message from that vehicle. All such vehicles are considered as neighbour vehicle. Here, two tables are maintained by each vehicle, they are Scheduling Table and Packet Delivery Ratio Table which are described below.

#### 3.1 Scheduling Table and Packet Delivery Ratio Table

The Scheduling Table which is maintained at every node consists of the following fields, *From Vehicle*, *Deadline*, *Remn\_Hops*, *Original Source*, *Flow ID*, *Priority* and *ReqSelPre*. *From Vehicle* is that vehicle from which the packet was obtained. *Deadline* refers to the intended end-to-end delay of the packet which is set by the original source. *RemnHops* gives the remaining number of hops to reach the destination. *ReqSelPre* gives the request selection precedence value of the packet. The format of the Scheduling table entry is revealed in the below expression.

$< \text{from\_vehicle}, \text{Deadline}, \text{Remn\_Hops}, \text{Original\_source}, \text{Flow\_ID}, \text{Req\_Sel\_Pre} >$

The packet delivery ratio table is used by the node to calculate the present packet delivery ratio of that flow. The Packet delivery ratio table includes *Original Source*, *Flow ID*, *Pkts Sent and Ack received*. In Packet delivery ratio table the Pkts Sent field provides the calculation of the packets flow which have been sent by the node if it is source or intermediate node or the calculation of the packets which have been received by the node if it is destination node. The Ack received field counts the number of acknowledgement

packets for that flow collected so far by that node. For the destination node the Ack received field is set as NULL. The packet delivery ratio table is used by the node to calculate the present packet delivery ratio of that flow. The format of the Packet delivery ratio table entry is revealed in the below expression.

*<Original\_source, Flow\_ID, Pkts\_Sent, AckReceived >*

### 3.2 Request Selection Precedence

Request Selection Precedence mechanism is introduced here to schedule the most appropriate request for service in the next cycle. Based on type of request priority and data priority the request selection precedence value is calculated. Depending on request priority, vehicles are categorized into three major groups as described in Table 1 and request data are categorized into three major groups as described in Table 2. The term request selection precedence (*ReqSelPre*) is defined using parameters request priority and data priority. It is defined as multiplication of request priority and data priority. The main objective of *ReqSelPre* is to ensure fairness in the scheduling process by targeting most appropriate request for service in the next cycle which is described by the equation 1.

**Table 1: Request's categorization**

Sl No	Req. Priority	Vehicle Type	Req. Value
1	High	Ambulance Medical Aid Police Fire-brigade Accident Vehicle	$\alpha_1$
2	Moderate	Abnormal Vehicle	$\alpha_2$
3	Low	Normal Vehicle	$\alpha_3$

**Table 2: Data categorization**

Sl No	Data <sub>prior</sub>	Data Type	Priority <sub>value</sub>
1	High	Ambulance Hospital Info Police Info Fire-brigade Info Accident Info	$\beta_1$
2	Moderate	Traffic Condition Road Condition Fuel Pumps Hotels Parking Info	$\beta_2$
3	Low	Commercial Info	$\beta_3$

$$ReqSelPre = \alpha_n * \beta_n, \quad \text{where } n = 1, 2, 3. \quad (1)$$

The vehicular ad-hoc network consists of three types of request priority vehicle  $\alpha_1, \alpha_2, \alpha_3$  and three types of data pri-

ority  $\beta_1, \beta_2, \beta_3$ . By multiply these two parameters *ReqSelPre* is calculated which ensure the fairness in the scheduling process. Let the value of request priority is assigned as  $\alpha_1 = 3, \alpha_2 = 2, \alpha_3 = 1$  and data priority as  $\beta_1 = 3, \beta_2 = 2, \beta_3 = 1$ . For example, if high priority vehicle  $\alpha_1$  wants to send high priority data  $\beta_1$ , then the value of *ReqSelPre* becomes  $3 * 3 = 9$ . These ensure if the high priority vehicle wants to send high priority data, it is served before the case where high priority vehicle wants to send low priority or moderate priority data as depicted in Table 3.

**Table 3: ReqSelPre Table**

	$\beta_1 = 3$	$\beta_2 = 2$	$\beta_3 = 1$
$\alpha_1 = 3$	9	6	3
$\alpha_2 = 2$	6	4	2
$\alpha_3 = 1$	3	2	1

### 3.3 Deadline Calculation

If node A wants to send a packet to node B through n number of links with transmission rate R bps. Suppose the nodes are separated by m meters and the propagation speed is s meters/sec. The deadline of the packet which is send by node A is  $n(\frac{m}{s} + \frac{L}{R})$  sec, where L is the size of the packet.

$$\begin{aligned} \text{The standard calculation of E - to - E Delay (D)} \\ = \sum (\text{Transmission Delay} + \text{Propagation Delay} \\ + \text{Processing Delay}). \end{aligned}$$

Here, processing delay has been ignored as nodes are not performing any complex encryption algorithm.

$$\begin{aligned} \text{Propagation delay} &= \frac{\text{m meters}}{\text{s meters/sec}} = \frac{m}{s} \text{ sec.} \\ \text{Transmission delay} &= \frac{\text{L bits}}{\text{R bits/sec}} = \frac{L}{R} \text{ sec.} \end{aligned}$$

Therefore, the deadline of a packet send from node A to node B through n number of links is set as  $n(\frac{m}{s} + \frac{L}{R})$  sec.

### 3.4 Queue Management

The transmission queue follows Little's algorithm [20] to transmit packet to the next node. Little's theorem is used in this system to meet delay target by controlling the packet arrival rate of the flow. The brief description of the Little's theorem is as follows:

$$\begin{aligned} \text{Average Delay of a packet (T)} &= \frac{N}{\lambda} \\ \text{where, } N &= \text{average number of packets in the queue} \\ T &= \text{average delay of a packet} \\ \lambda &= \text{packet arrival rate} \end{aligned}$$

## 4. PROPOSED DATA DIFFUSION METHOD

In this paper a priority based exigent data dissemination approach has been proposed where a laxity based priority scheduling technique with back-off mechanism is introduced. The next packet is to be transmitted based on its priority considering the laxity budget of the packet, the present packet delivery ratio, required packet delivery ratio and the request selection precedence. Request selection precedence is introduced to provide the fairness in the scheduling process which services the most urgent request in the next iteration. The present packet delivery ratio (PDR) of a route is

present in the nodes scheduling table (ST), described in section 3, which is a significant parameter used for calculating the precedence of a packet belonging to that route with respect to packets belonging to other routes that have entries in the ST of the node. For this reason, a node needs to maintain track of the PDR of all routes that have accesses in the ST. This is accomplished particularly by a feedback method.

Functionality of feedback method is depicted in Fig.1 . Incoming packets are first fed into an input-queue. According to their request selection precedence (ReqSelPre) value and the laxity information of the packet, the scheduler schedules them and feed them into the transmission queue. The transmission queue follow Little's algorithm as described in section 3.4. After transmitting a packet, the node keeps track of the information retained in its packet delivery ratio table concerning the number of packets conveyed so far. On receiving data packets, the destination node commences a feedback by means of which information about the number of packets received by it is transmitted to the source. The feedback information handler received these two information which is denoted by  $S_i$ . As well as, the feedback information handler sends the previous state information  $S_{i-1}$  to the priority function unit.

The laxity budget information which is stored in the Node's scheduling table is also sent to the priority function unit. These two information is used to calculate the priority indices of the packets. After transmitting a packet successfully each node increments the value of pktsSent field as well as after receiving a packet successfully by the destination node, the destination node update the packet received field. The key purpose of the feedback mechanism is to get the present packet delivery ratio of each packet of each stream. The packet delivery ratio (PDR) is calculated by  $PDR = \frac{\text{Ack received}}{\text{Pkts Sent}}$ . The priority index of a packet is a pointer representing the priority of the packet. Here, packet's priority is inversely proportional to the priority index.

$$\text{Packet Priority} \propto \frac{1}{PI} \quad (2)$$

The priority index of a packet is defined as below:

$$PI = \frac{PDR * ULB}{DPDR} \quad (3)$$

where,  $PI$  is the priority index of the packet,  $PDR$  is the packet delivery ratio.  $DPDR$  is the desired packet delivery ratio.  $ULB$  is the Uniform Laxity Budget of the packet.

Uniform Laxity Budget (ULB) of the packet is defined as follows:

$$ULB = \frac{\text{Slack-time} * \text{remaining hops}}{\text{ReqSelPre}} \quad (4)$$

where,  $Slack-time$  is the time difference between dead-line (described in section 3.3) and present time,  $ReqSelPre$  is request selection precedence value which is described in section 3.2. The MAC layer of the source vehicle is responsible to provide the value of the remaining hops. Here, the MAC layer of the source vehicle applies an individual interface task by which the MAC layer request the network layer for the remaining number of hops to be passed by the packet.

As per the above equation the priority index composed of two important expressions  $PDR/DPDR$  and  $ULB$ . The expression  $PDR/DPDR$  gets high value when more number of packets of a stream meet their delay target. Therefore for packets belonging to that stream, the priority of the packets becomes lower as the priority index gets high value. When a very small number of packets of a stream meet their delay targets, the  $PDR/DPDR$  value become very less, by this means lowering the value of priority index and raising the priority of packets belonging to that stream. For calculating the priority index, uniform laxity budget( $ULB$ ) also plays a very vital role. There are two parameters in  $ULB$ ; one of them is number of hops remaining to be traversed, when a packet is near its original source and needs to move numerous hops to reach the destination, its priority index value will be higher, thus lowering the priority of the packets. When the packet near its destination, the less number of hops to be traversed tends to decrease the priority index, thereby the priority of the packet is higher. This method helps the packets of high speed vehicle's to reach its destination before it going beyond its transmission range of the destination. Another important parameter of  $ULB$  is request selection precedence ( $ReqSelPre$ ). When the value of  $ReqSelPre$  is higher the value of  $ULB$  becomes less so the value of priority index becomes less and as a result the actual priority of the packet becomes high. Means those packets whose  $ReqSelPre$  value is high, they are served before those packets who has less  $ReqSelPre$  value. These helps to provide the fairness in the scheduling method which services the most urgent request in the next iteration.

## 4.1 Back-off policies

Back-off policy ensures the process of prioritization of the waiting data packets in the broadcast region based on some criteria as described in the following expression. If a node contains highest priority data packets, it gains the lowest back-off time and vice-versa. Thus the back-off policy upholds the following scheme. Let  $r_v$  be the rank of vehicle  $v$ 's packet in its own scheduling table,  $l$  is the number of retransmission slots made for the packet,  $l_{max}$  is the maximum number of retransmission slots allowed, then the back-off interval is given by

$$\text{back-off} = \begin{cases} \text{unif}[0, 2^l CW_{min} - 1], & \text{if } r_v=1 \text{ and } l < l_{max} \\ \frac{PDR}{DPDR} \times CW_{min} + \text{unif}[0, CW_{min} - 1], & \text{if } r_v > 1 \text{ and } l = 0 \\ ULB \times CW_{min} + \text{unif}[0, l^2 CW_{min} - 1] & \end{cases}$$

where,

$CW_{min}$  is the minimum size of the contention window,  
 $PDR$  is the packet delivery ratio,  
 $DPDR$  is the desired packet delivery ratio, &  
 $ULB$  is the uniform laxity budget of the packet.

The first expression of back-off policy describe that the highest ranked packet has the lowest back-off period. The second expression describe the back-off period of that packet

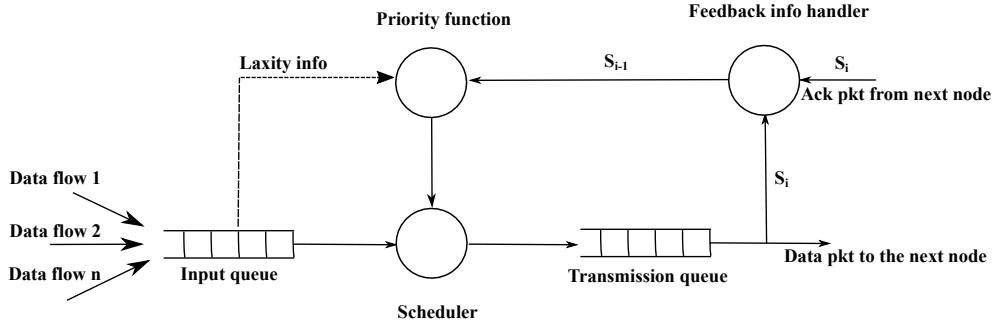


Figure 1: Feedback Mechanism

which is not the highest ranked packet but it is the first time the packet is being conveyed. At last if the packet does not well into these two class, then the back-off period of that packet is described by the third expression.

## 5. PERFORMANCE EVALUATION

### 5.1 Simulation model and parameter set-up

EXataCyber-2.0 Network Simulator has been used to integrate the proposed work with existing protocols. First of all, the physical layer radio type has been considered as IEEE 802.11p [11] [15]. The simulation area 1000m x 1000m with different vehicle speed 25km/h, 30km/h, 35km/h, 40km/h and 45km/h is chosen. Data rate has been fixed to 9 Mbps with three different packet size 256bytes, 512 bytes and 1024 bytes. Three different network scenarios with the following parameters is considered: omni directional antenna model, two ray pathloss model, lognormal shadowing model and ricean fading model with channel frequency 5.9 GHz. The temperature as 290k and SNR threshold value as 4dBm is fixed as shown in Table 4 and Table 5 describe the Ricean fading model.

Table 4: System parameters used

Parameter	Value
Terrain	1000m X 1000m
Topology	Urban
Data Rate	9 Mbps
No of Vehicles	50, 70, 90, 110
Packet Size	256,512,1024 (bytes)
Vehicle Velocity	25,30,35,40,45(km/h)
Traffic Type	CBR
Physical Layer Radio	IEEE 802.11p
Antenna Model	Omni directional
Temperature	290 K
SNR Threshold	4 dBm
Channel Frequency	5.9 GHz
Pathloss model	Two Ray
Shadowing Model	Lognormal
Fading Model	Ricean
Simulation Time	500 Sec

Here, two ray path-loss model [18] is used which considered both the direct path and a ground reflection path to

Table 5: Ricean Fading Model Properties

Parameter	Value
Maximum velocity (m/s)	10.0
Signal propagation speed (m/s)	$3e^8$
Propagation limit (dBm)	-111.0
Proximity	400

accurate prediction at a long distance than the free space model. Further the Log normal Shadowing model is used as this is appropriate for the proposed scenario which consider vehicle mobility and obstructions of the urban environment.

The User Datagram Protocol (UDP) is used as the transport layer protocol and in the application layer Constant Bit Rate (CBR) data traffic is applied between source and destination with different packet size 256 bytes, 512 bytes and 1024 bytes. Here, the vehicle-to-vehicle radio link can be formed statistically as a Ricean fading model [19]. The dominant module in the Ricean fading channel is expected to be comparatively strong contrasted to the reflected signal and the delay increase is expected to be comparatively small as reflections take place in the instant environs of the transmitter and the receiver antenna. The dissemination channel is formed as a dominant factor consisting of a direct-line-of-sight signal and a ground reflected signal, a set of early reflected signals and the inter symbol intrusion caused by exceptionally overdue signals.

*Scenarios:* In the simulation model three different types of scenarios is considered broadly. All the scenarios are tested in urban environment.

- In the first type, the scenarios are considered varying different vehicle density. The scenario consists of 50 vehicles, 70 vehicles, 90 vehicles and 110 vehicles.
- In the second type, the scenarios are considered varying different vehicle speed from 25km/h to 45km/h.
- In the third type, the scenarios are considered varying different packet size from 256 bytes to 1024 bytes.

### 5.2 Simulation metrics

The following six performance metrics are considered in this simulation study:

- **Packet Delivery Ratio (PDR):** Average packet delivery ratio represents the average ratio of the number of successfully received data packets at the destination node to the number of data packets sent [15] [16]. PDR is calculated as:

$$PDR = \frac{\sum \text{Number of packet received}}{\sum \text{Number of packet send}} \times 100$$

- **Throughput (Thr):** Throughput is defined as the average data transfer rate in Mbps measured between a source and destination. Throughput is adversely affected due to network congestion and signal interference from other nearby networks [14]. The simulation result for throughput in EXataCyber-2.0 shows the total received packets at destination in Mbps, mathematically throughput is calculated as:

$$\text{Thr} = \frac{\sum \text{number of packet received at destination}}{\sum \text{Simulation time}} \times \text{Packet Size}$$

- **End-To-End Delay (E – to – E Delay):** The end-to-end delay is the time taken by a packet to move from source to destination node and is calculated by the elapsed time when a packet is sent by the source node to the time when it is received at the destination node. This includes all possible delays like buffering, queuing and processing at intermediate routers [17]. The result of End-to-End Delay is measured in millisecond. End-to-End Delay is calculated as:

$$\text{E – to – E Delay} = \sum(\text{Transmission Delay} + \text{Propagation Delay} + \text{Processing Delay})$$

- **Jitter (Jitter):** Jitter is defined as the variation of a signal's instants from their ideal positions in time i.e. the deviation from the ideal timing of an event. Jitter arises due to the variation of packet delay i.e. the variation in time between packets arriving, caused by network congestion or route changes. In a communication system, the accumulation of jitter will eventually lead to data errors. Jitter is measured in millisecond.

- **Packet Retransmission (Pkt retrans):** Number of packets rescheduled due to lack of acknowledgement (ACK) response to packet transmission due to time out of the packet interval.
- **Packet Drop (Pkt drop):** Number of packets not rescheduled after exceeding the maximum number of unsuccessfully delivered fragments.

## 5.3 Result Analysis

### 5.3.1 Effect of node density

In this set of simulation, the vehicle density is varied by varying the number of vehicles, remaining the terrain area unchanged. The terrain dimension chosen here is 1000m x 1000m and the maximum vehicle speed is 25km/h. Fig. 2 shows the variation of packet delivery ratio with varying the number of vehicles for the proposed scheme-adapted distributed laxity based priority scheduling scheme which is coined as ADLPS and the 802.11 DCF. The proposed scheme is outperform than the 802.11 DCF. Fig. 3 shows the average throughput which is also better perform than 802.11 DCF. As the number of node increases, the contention for the channel also increased and for these the nodes

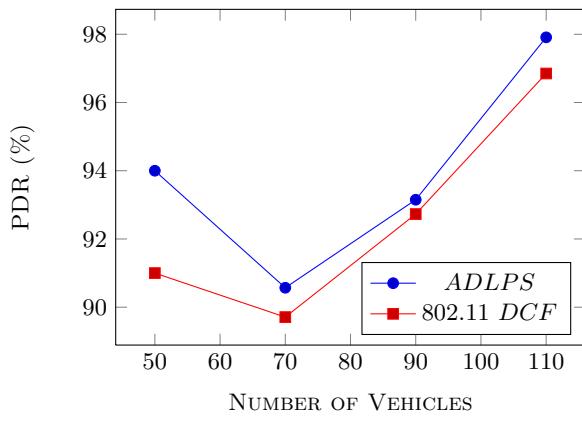
need to wait for a long times before transmitting the packets. This causes an increase in average end-to-end delay of 802.11 DCF schemes. But in the proposed scheme the back-off mechanism is introduced. The back-off mechanism considered the uniform laxity budget (ULB). Therefore the average end-to-end delay and jitter of the proposed system remains low due to timeliness of the packets as depicted in Fig. 4 and Fig. 5. Fig. 6 and Fig. 7 describes the number of packet retransmission and the number of packet drop respectively with varying the number of vehicles. In respect of packet retransmission and packet drop, the proposed system is better perform than the existing protocol 802.11 DCF. Packets priority assignment supported on flow characteristics along with a back-off mechanism that guarantees channel contention based on these priorities, facilitates our method to execute better.

### 5.3.2 Effect of node velocity

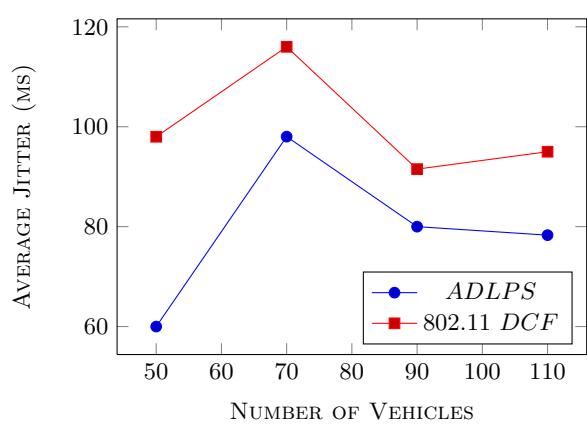
In this set of simulations, the speed of the vehicles is varied from 25km/h to 45km/h, remaining the terrain area unchanged and the number of vehicles remain unchanged at 50. The terrain dimension chosen here is 1000m x 1000m. Fig. 8 shows the variation of packet delivery ratio with varying the number of vehicles for the proposed scheme-adapted distributed laxity based priority scheduling scheme ADLPS and the 802.11 DCF. The proposed scheme is outperform than the 802.11 DCF. Fig. 9 shows the average throughput which is also better perform than 802.11 DCF. Average end-to-end delay and average jitter is also perform well of the proposed scheme than 802.11 DCF depicted in Fig. 10 and Fig. 11. Fig. 12 and Fig. 13 describes the number of packet retransmission and the number of packet drop respectively with varying the number of vehicles. Number of packet retransmission and packet drop in the proposed system is better perform than the existing protocol 802.11 DCF. Thus, even when the node velocity is varied, our scheduling method facilitates consistently well, in so doing better result.

### 5.3.3 Effect of packet size

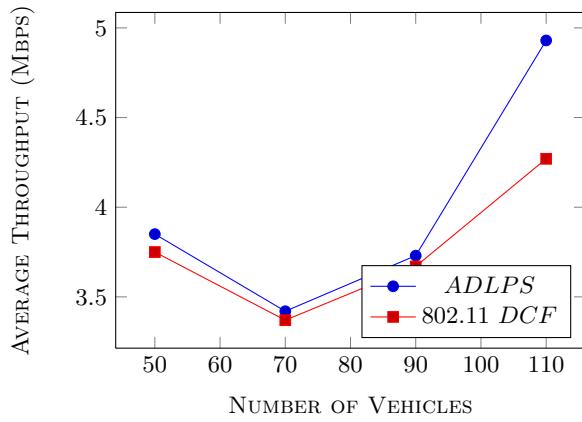
In this set of simulations, the packet size is varied from 256 bytes to 1024 bytes, remaining the terrain area unchanged and the number of vehicles remain unchanged at 50. The terrain dimension chosen here is 1000m x 1000m. Fig. 14 shows the variation of packet delivery ratio with varying the number of vehicles for the proposed scheme-adapted distributed laxity based priority scheduling scheme ADLPS and the 802.11 DCF. The proposed scheme is outperform than the 802.11 DCF. Fig. 15 shows the average throughput which is also better perform than 802.11 DCF. Average end-to-end delay and average jitter is also perform well of the proposed scheme than 802.11 DCF depicted in Fig. 16 and Fig. 17. Fig. 18 and Fig. 19 describes the number of packet retransmission and the number of packet drop respectively with varying the packet size. In respect of packet retransmission and packet drop, the proposed system is better performs than the existing protocol 802.11 DCF. From the above graphs it can be seen that increasing packet size does not influence our scheduling method, which hence achieves better than the existing technique.



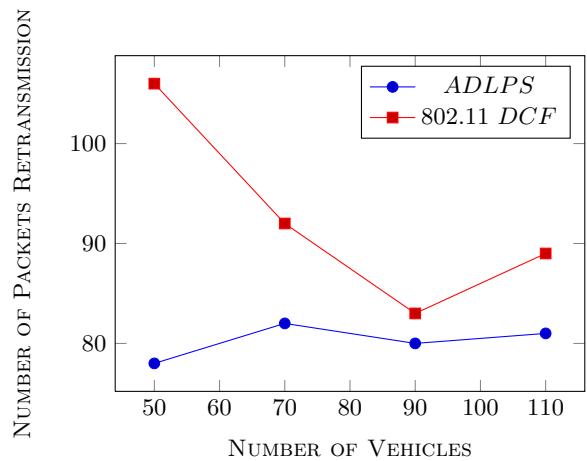
**Figure 2: Packet Delivery Ratio vs number of Vehicles**



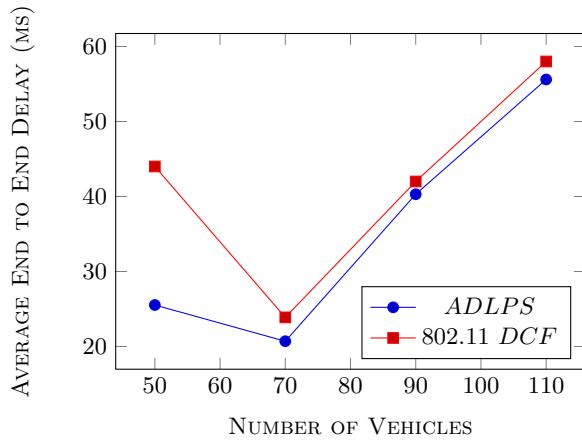
**Figure 5: Average Jitter vs number of Vehicles**



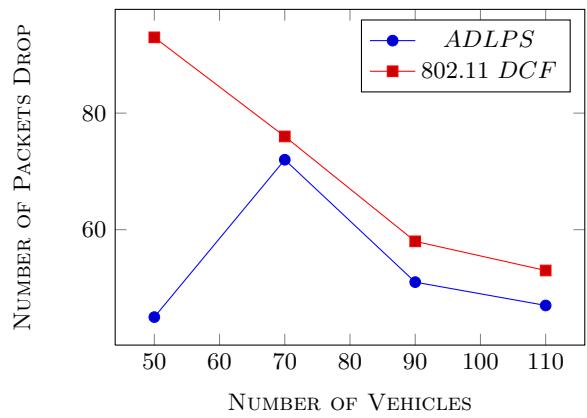
**Figure 3: Average Throughput vs number of Vehicles**



**Figure 6: Number of packets retransmission vs number of Vehicles**



**Figure 4: Average End-to-End Delay vs number of Vehicles**



**Figure 7: Number of packets drop vs number of Vehicles**

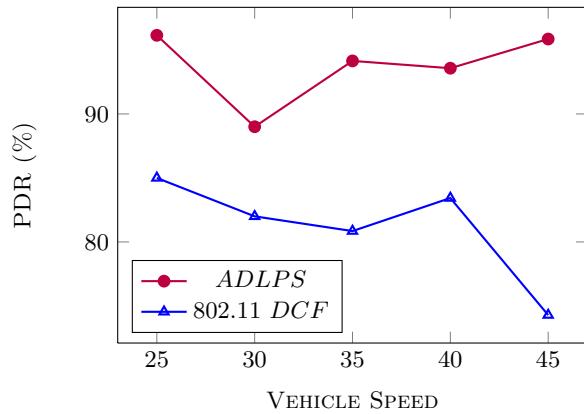


Figure 8: Packet Delivery Ratio vs Vehicle Speed

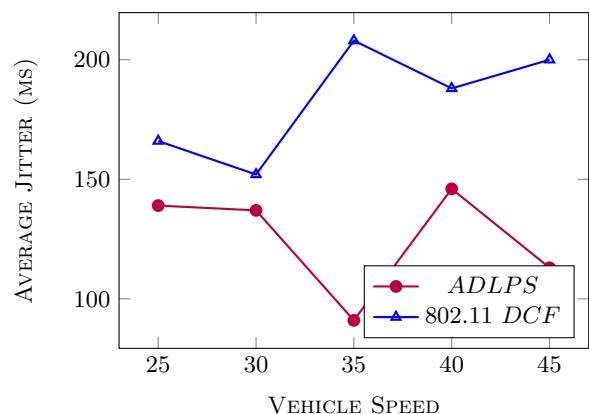


Figure 11: Average Jitter vs Vehicle Speed

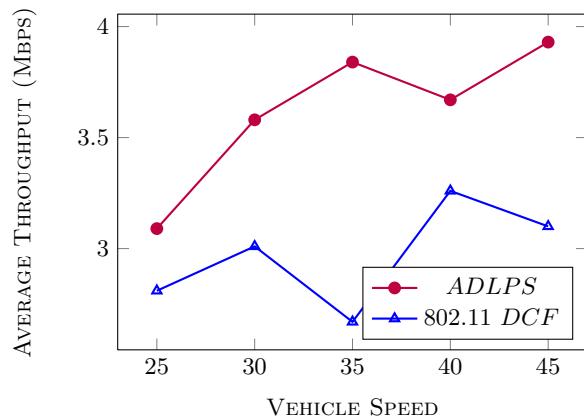


Figure 9: Average Throughput vs Vehicle Speed

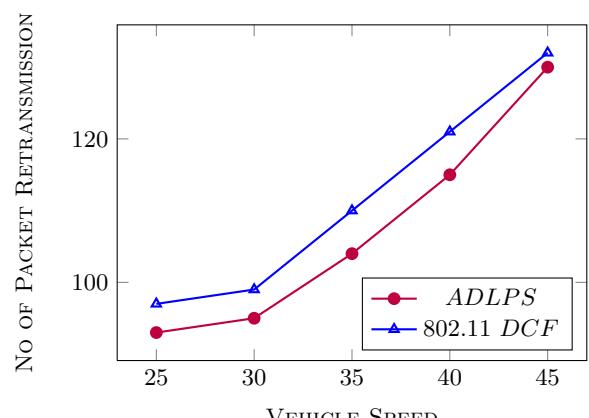


Figure 12: Number of packets retransmission vs Vehicle Speed

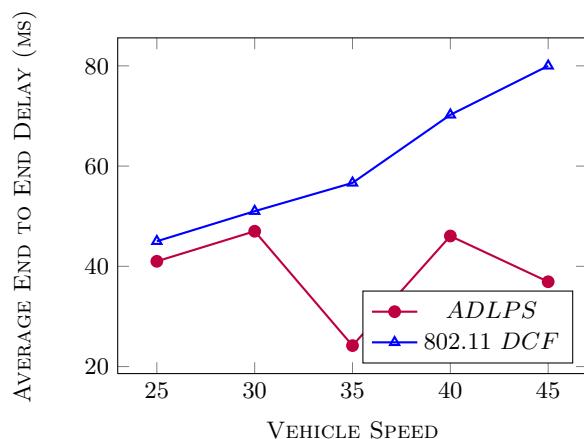


Figure 10: Average End-to-End Delay vs Vehicle Speed

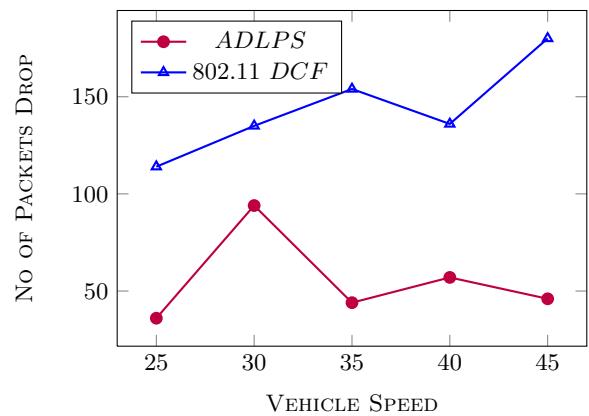


Figure 13: Number of packets drop vs Vehicle Speed

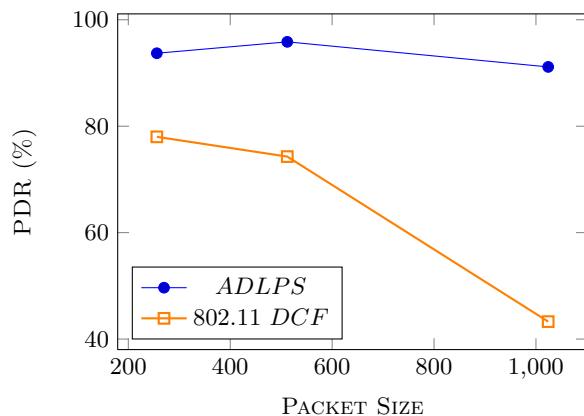


Figure 14: Packet Delivery Ratio vs Packet Size

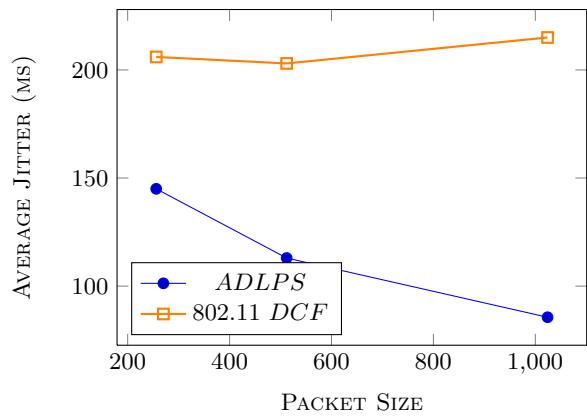


Figure 17: Average Jitter vs Packet Size

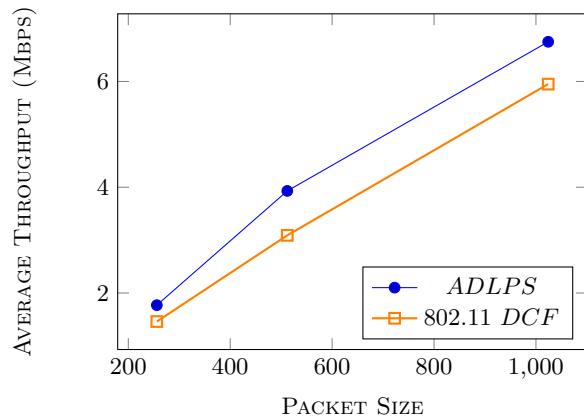


Figure 15: Average Throughput vs Packet Size

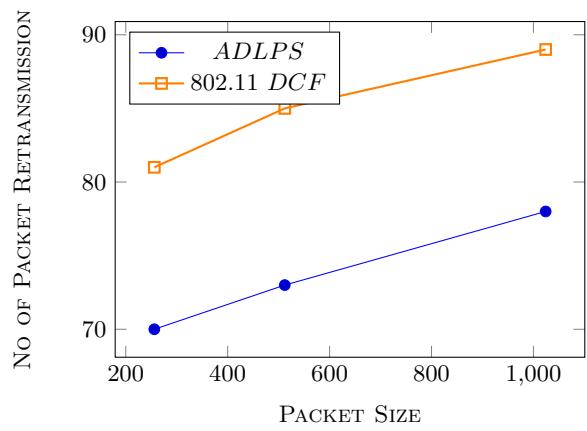


Figure 18: Number of packets retransmission vs Packet Size

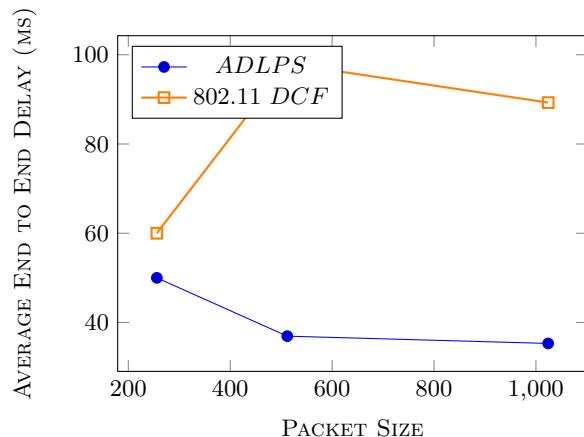


Figure 16: Average End-to-End Delay vs Packet Size

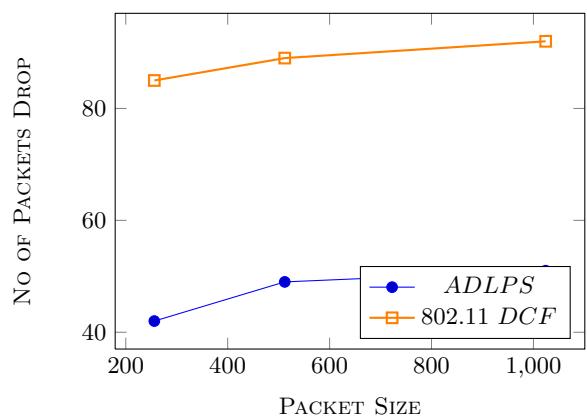


Figure 19: Number of packets retransmission vs Packet Size

## 6. CONCLUSIONS

In this paper a priority based exigent data dissemination approach has been proposed for time sensitive traffic of VANet where a laxity based priority scheduling technique with back-off mechanism is introduced. The scheme selects the packet to be transmitted next, based on a priority calculation function that takes into consideration the uniform laxity budget of the flow, current packet delivery ratio of the flow, the desired packet delivery ratio and the request selection precedence value. Here a back-off scheme is also introduced, where a vehicle varies its back-off period depending on the rank of its highest precedence packet, with esteem to other such high precedence packets queued at nodes in its locality. By means of simulation studies, the performance of the proposed scheme is performed better with that of the IEEE 802.11 DCF.

## 7. ACKNOWLEDGMENTS

The author would like to thank University Grants Commission, Government of India for providing financial support.

## 8. REFERENCES

- [1] Jagruti Sahoo, Eric Hsiao-Kuang Wu, Pratap Kumar Sahu, and Mario Gerla, (2011) "Binary-Partition-Assisted MAC-Layer Broadcast for Emergency Message Dissemination in VANETs", IEEE Transactions on Intelligent Transportation Systems, Vol. 12, No. 3, September 2011.
- [2] P.A.Sumayya a, P.S.Shefeena. (2015) "VANET Based Vehicle Tracking Module for Safe and Efficient Road Transportation System", International Conference on Information and Communication Technologies (ICICT 2014), Procedia Computer Science 46, pp.1173–1180, doi: 10.1016/j.procs.2015.01.030.
- [3] Kai Liu, Joseph K. Y. Ng, Victor C.S. Lee, Sang H. Son and Ivan Stojmenovic. (2015) "Cooperative Data Scheduling in Hybrid Vehicular Ad Hoc Networks: VANET as a Software Defined Network", IEEE/ACM Transactions on Networking, pp.1–15, doi: 10.1109/TNET.2015.2432804.
- [4] Nan Cheng, Ning Lu, Kai Liu, Xinhong Wang, and Fuqiang Liu. (2010) "A Prioritized Resource Scheduling Scheme for Throughput-sensitive Applications in VANET", 978-1-4244-3709-2/10/2010 IEEE.
- [5] Masato Nakamura, Tomoya Kitani†, Weihua Sun, Naoki Shibata††, Keiichi Yasumoto, and Minoru Ito. (2010) "A Method for Improving Data Delivery Efficiency in Delay Tolerant VANET with Scheduled Routes of Cars", 978-1-4244-5176-0/10/2010 IEEE.
- [6] Guillaume Demesure, Michael Defoort, Abdelghani Bekrar, Damien Trentesaux and Mohamed Djemai. (2015) "Navigation Scheme with Priority-Based Scheduling of Mobile Agents: Application to AGV-Based Flexible Manufacturing System", J Intell Robot Syst, doi: 10.1007/s10846-015-0273-4.
- [7] Zhang, Y., Zhao, J. and Cao, G. (2010). "Service scheduling of vehicle-roadside data access". In Springer Science Mobile Network Application (pp. 83–96).
- [8] Zhang, Y., and Cao, G. (2011). "V-PADA: Vehicle platoon aware data access in VANETs". IEEE Transactions on Vehicular Technology, 60(5), pp. 2326-2339.
- [9] Yen, Y.-S., Chao, H.-C., Chang, R.-S., and Vasilakos, A. (2011). "Flooding-limited and multi-constrained QoS multicast routing based on the genetic algorithm for MANETs". Mathematical and Computer Modelling, 53(11-12), 2238-2250.
- [10] Li, P., Guo, S., Yuy, S., and Vasilakos, A. V. (2012). "CodePipe: An opportunistic feeding and routing protocol for reliable multicast with pipelined network coding". INFOCOM, 2012, 100-108.
- [11] Meng, T., Wu, F., Yang, Z., Chen, G., and Vasilakos, A. V. (2015). Spatial reusability-aware routing in multi-hop wireless networks. IEEE TMC,. doi:10.1109/TC.2015.2417543.
- [12] Zeng, Y., Xiang, K., Li, D., and Vasilakos, A. V. (2013). Directional routing and scheduling for green vehicular delay tolerant networks. Publication of Wireless Networks, 19(2), 161-173.
- [13] Zhou, L., and Vasilakos, A. V. (2011). Distributed media services in P2P-based vehicular networks. IEEE Transaction of Vehicular Technology, 60(2), 692-703.
- [14] Y. Chang, C.P. Lee, and John A. Copeland. (2011). "Goodput Enhancement of VANETs in Noisy CSMA/CA Channels". In IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 29, NO. 1, JANUARY 2011.
- [15] A. Abdrabou and Weihua Zhuang. (2011). "Probabilistic delay control and road side unit placement for vehicular ad hoc networks with disrupted connectivity". Selected Areas in Communications, IEEE Journal on, 29(1):129–139, January 2011. ISSN 0733-8716. doi: 10.1109/JSAC.2011.110113.
- [16] J.A.P. Martins, S.L.O.B. Correia, and J. Celestino. (2010). "Ant-dymo: A bio-inspired algorithm for manets". In 17th International Conference on Telecommunications, 2010, pages 748–754, April 2010. doi:10.1109/ICTEL.2010.5478808.
- [17] X. Wang, C. Liu, Y. Wang, and C. Huang. (2014). "Application of ant colony optimized routing algorithm based on evolving graph model in vanets". In 17th International Symposium on Wireless Personal Multimedia Communications (WPMC2014), pages 265-270, Sept 2014. doi: 10.1109/WPMC.2014.7014828.
- [18] C. Sommer, S. Joerer and F. Dressler. (2012). "On the Applicability of Two-Ray Path Loss Models for Vehicular Network Simulation". In 4th IEEE Vehicular Networking Conference (VNC 2012), DOI: 10.1109/VNC.2012.6407446.
- [19] Maja Ilic-Delibasic and Milica Pejanovic-Djurisic. (2012). "A Novel Ricean Channel Fading Model With Random K Factor". In Wireless Telecommunications Symposium (WTS), 18-20 April 2012, London, DOI: 10.1109/WTS.2012.6266125.
- [20] Song-Hee Kim and Ward Whitt. (2013). "Statistical Analysis with Little's Law". In Operations Research Vol. 61, No. 4, July–August 2013, pp. 1030–1045, ISSN 0030-364X (print), ISSN 1526-5463 (online), <http://dx.doi.org/10.1287/opre.2013.1193>.

# An Energy-efficient and Buffer-aware Routing Protocol for Opportunistic Smart Traffic Management\*

Sobin CC  
Dept. of CSE, IIT Roorkee  
Utharkhand, India  
sobincc@gmail.com

Vaskar Raychoudhury  
AvH Fellow,  
University of Mannheim  
Germany  
vaskar@ieee.org

Snehanshu Saha  
PESIT-South Campus  
Banglore, India  
snehanshusaha@pes.edu

## ABSTRACT

Smart Traffic Management (STM) is a major application domain for developing Smart City systems. In an STM, sensors attached to the vehicles sense the environment and exchanges the sensed data with other vehicles. Due to the high rate of mobility, an STM suffers from frequent disconnections and need to resort to opportunistic encounters for communication. Since the sensors used in STM are having limited energy and buffer space, designing energy-efficient and buffer-aware message forwarding is quite challenging in an opportunistic STM. In this paper, we have designed an energy-efficient and buffer-aware routing protocol, EBR, for an opportunistic STM, which will select the relay nodes based on their remaining energy level and buffer space. We have conducted extensive simulations for evaluating the performance of our proposed EBR protocol. We also have developed a prototype smart vehicular test-bed for evaluating performance of the EBR, in real time. Both simulation and test-bed results show that the EBR outperforms some of the existing opportunistic routing protocols in terms of delivery ratio, delivery delay and energy efficiency.

## Keywords

Smart City, Smart Traffic Management, Energy

## 1. INTRODUCTION

With the introduction of smart cities, Information and Communication Technologies (ICT) play an important role to improve the quality of life with various smart solutions. Smart Traffic Management (STM) system is one such smart solution. An STM is one, in which the vehicles are equipped with sensors (accelerometer, light sensor, proximity sensor, etc.) which helps them sense the surrounding environment and to take decisions based on that (Fig.1). The sensors

\*(Produces the permission block, and copyright information). For use with SIG-ALTERNATE.CLS. Supported by ACM.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ICDCN '17, January 04-07, 2017, Hyderabad, India

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007757>

help to analyze the traffic pattern in the city, notify accidents and over speeding, provide information about available parking slots, etc. Since, the sensors used in STMs are energy-constrained, operation of an STM face several non-trivial challenges as mentioned below.

- Nodes are energy-constrained
- Communication is energy consuming
- Topology is unstable, so message redundancy will be necessary

Also, due to the fast mobility of the vehicles in an STM, there are chances of frequent disruptions in inter-vehicular connectivity, which will necessitate message re-transmission demanding even further energy. So, operating an STM system is extensively energy-demanding and if the sensing and actuating devices are self-powered (not powered by the vehicle), there are several non-trivial issues that require to be addressed for efficient energy management. Nowadays many law enforcement agencies of the world are looking for energy-harvesting sensors [1][2][3] to be used in STM system, which will not be powered by the vehicle and thus will not be in control of the car owner/driver.



Figure 1: Example of an STM

Opportunistic communication [4] can be used in an environment where an end-to-end connectivity does not exist. Participating nodes in an opportunistic network follows store-carry-and forward mechanism for message forwarding. Since, messages are exchanged from source to destination in

a hop-by-hop manner through multiple relay nodes, selection of appropriate relay node influences the routing performance. Hence, most of the researchers [5][6][7][8][9] focused on relay selection in opportunistic routing. Please refer to [10] for a recent state-of-the-art of routing and data dissemination in opportunistic networks.

However, energy issues associated with message routing in opportunistic networks have grossly been overlooked in the existing literature. But, we observed through simulation as well as real-time experiments that, the energy as well as resource constraints of the nodes (sensors in an STM) also influences the relay selection, which in turn affects routing performance. Suppose, a node with limited energy as well as buffer space is selected as a relay, it will not forward the message further in the network, which will degrade the routing performance such as reducing the delivery ratio and increasing the delivery delay (Refer to section 5.1).

In this paper, we propose an energy-efficient routing protocol, namely Energy-efficient and Buffer-aware Routing (EBR) for an opportunistic STM. The energy efficiency is achieved by reducing the number of message transmissions in the network by selecting only those nodes as relays, which are having sufficient energy level and buffer space to forward the message further in the network. This is because, irrespective of the heuristics used for relay selection, capability of the relay node is the most important factor to be considered, as it will directly affect the routing performance. We have defined the capability of the node as a function of its remaining energy level and buffer space. We have verified through both simulations as well as real test-bed analysis that our EBR protocol improves the routing performance compared to existing opportunistic routing protocols.

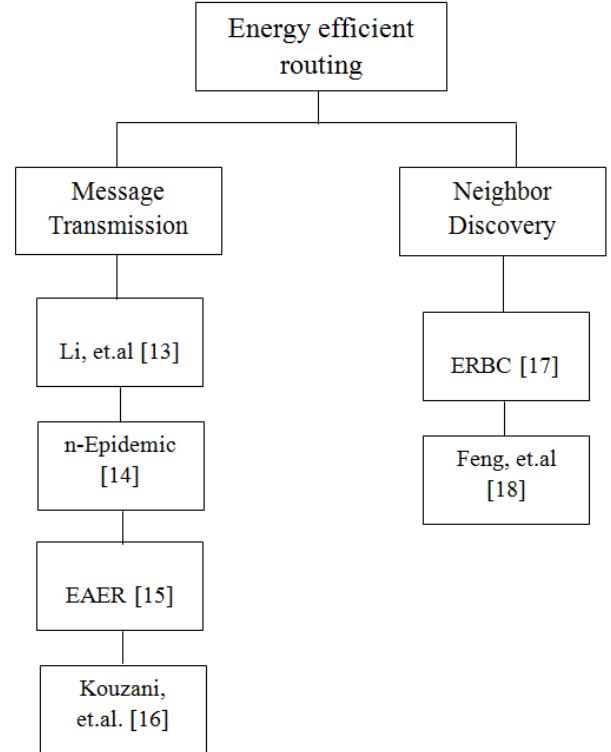
In summary, we have made the following contributions in this paper.

- We propose EBR, a novel energy-efficient and buffer-aware routing protocol for an opportunistic STM.
- We have conducted extensive simulations to evaluate the performance of our proposed EBR routing protocol. The results show that EBR achieves a higher delivery ratio, lower delivery delay and higher energy efficiency compared to existing opportunistic routing protocols.
- As a proof of concept, we have developed a GPS-based vehicle tracking system over a smart vehicular test-bed, where cars communicate opportunistically with each other to share their location information. We have evaluated EBR protocol on test-bed and the results corroborate the simulation results.

## 2. RELATED WORKS

In this section, we will revisit some of the existing opportunistic routing protocols, which consider energy constraints. Since message transmission is the most important source of energy consumption, some of the researchers focused on reducing the number of message transmissions in order to save energy. Apart from that, there is some evidence of scholarly work on reducing energy consumption in the neighbor discovery phase. These include, but are not limited to, controlling the transmission power, proper selection of detection range and frequency, etc.

Initial efforts to study the energy consumption in opportunistic network have been made by Silva, et. al. [11], by contributing an energy module to Opportunistic Network Environment (ONE) [12] simulator. They have observed



**Figure 2: Classification of energy-efficient routing in opportunistic networks**

that because of the frequent node mobility and message forwarding among nodes, the energy associated with each node is drastically reduced and most of opportunistic application scenarios prevent nodes to get recharged. Li, et. al. [13], proposed an energy-efficient opportunistic forwarding for DTNs, in which the probability of message transmission is controlled using either static or dynamic forwarding policies. In case of static forwarding policy, the probability of message transmission is known in advance and does not change with time, whereas in case of dynamic forwarding policy, it changes dynamically.

An energy-efficient n-epidemic routing protocol have been proposed by Lu, et. al. [14], in which a message is transmitted by a node, only if, it has at least  $n$  neighbors. Rango, et. al. [15], have proposed an extension to n-epidemic, called as, Energy Aware Epidemic Routing (EAER). In EAER, instead of choosing the value of  $n$ , statically as in [14], the value is chosen dynamically based on either the current energy level of a node or its current neighbor density. Kouzani, et. al. [16], have modeled the trade-off between energy consumption and forwarding efficiency as an optimal control problem. They have proved through simulation that the optimal forwarding decisions follow the forwarding threshold of each node, which depends on the remaining energy in a node.

The other category of opportunistic routing protocols focused on reducing energy consumption in the neighbor discovery phase. Yao, et al. [17], focused on reducing the en-

ergy consumption based on power control mechanism. They have proposed an energy-efficient routing protocol, called as ERBC, in which the transmission power of a node is chosen dynamically, using cross-layer design and RSSI technique. Feng, et. al. [18], considered a different aspect of energy consumption based on the detection frequency and detection range. They have observed through simulation that the energy consumption can be reduced by properly selecting the detection range and detection frequency during the neighbor discovery phase.

In summary, most of the existing opportunistic routing protocols, which consider energy constraints, focus on reducing overall energy consumption of the network either in message transmission phase or in the neighbor discovery phase. None of them focus on identifying the capability of the nodes based on their energy level and buffer space. Suppose, if a message is forwarded to a neighbor node, which is having a low energy level and buffer space, it will not forward the message further in the network, so the message will not reach the destination.

In this paper, we have addressed the aforementioned issues and developed an energy-efficient and buffer-aware routing protocol, EBR, for an opportunistic STM. We found experimentally that, reducing the number of message transmissions is most suitable for an opportunistic STM, due to the high-speed of vehicles as well as the controlled flooding based routing strategy followed in message transmission.

### 3. SYSTEM MODEL

In this section, we will describe the assumptions and the problem description for the proposed EBR routing protocol.

#### 3.1 Network model

We assume there are  $N$  nodes in the network. Initially, each node is having fixed energy ( $E$ ). We assume that nodes communicate, when they are in the transmission range of each other. We assume that the energy level of a node will be discharged based on neighbor discovery, message reception and message forwarding. We assume that nodes cannot recharge their batteries once it drained out. The inter-contact time between any two nodes is exponentially distributed. We assume that each node in the network is having a fixed size buffer space of size  $B$  bytes. The contact duration among nodes and bandwidth is limited. All the messages are of the same priority and each message is having a fixed Time To Live value ( $TTL$ ). We have used a multi-copy message forwarding strategy. The underlying communication paradigm used is unicast.

#### 3.2 Mobility model

Since nodes in an STM use store-carry and forward paradigm for delivering messages from source to destination, the movements of nodes play an important role in the routing performance. Initially, each node is having fixed energy and buffer space available. The message is also associated with a fixed TTL value initially. On expiring TTL, the message is discarded from the node's buffer. Node mobility pattern decides the way in which a node discovers its neighbor nodes, which is having an impact on energy consumption of the node as well as the message delivery ratio, delay and energy efficiency. We assume following requirements for our mobility model.

- A node moves randomly and independently of other nodes
- The meeting rate among any two nodes follow an exponential distribution.

The first requirement will satisfy the requirements of a smart vehicular environment, where vehicles move independently and randomly with a high-speed. Due to the random movement and high-speed of the vehicles, the inter-meeting time between two vehicles is exponentially distributed.

#### 3.3 Problem definition

Based on the assumptions discussed earlier, we now provide our formal problem definition. In a network of  $N$  mobile nodes, consider the presence of a source ( $S$ ) and destination ( $D$ ) node pair, which communicates through a set  $R \in N$  of relay nodes *via* opportunistic contacts. The objective is to develop a routing protocol, for STM, having high delivery ratio, low delivery delay and high energy efficiency.

### 4. EBR PROTOCOL

In EBR, a node can locally decide from its neighbor nodes, whether a particular node has the capability to forward the message further in the network. When two nodes are in the transmission range of each other, the information about the remaining energy level and buffer space are exchanged with each other along with the summary vectors. After receiving such information, the source node will select a node as relay from the set of its neighbor nodes, if the remaining energy level and buffer space of the corresponding node is greater than a threshold. This relay selection strategy ensures that only those nodes, which are having a sufficient energy level and buffer space are participating in the message delivery.

Suppose, when the node  $i$  encounters node  $j$ , both  $i$  and  $j$  exchange the summary vectors ( $SV$ ) with each other, which contains the list of stored messages, remaining energy level and buffer space in each node. Then node  $i$  calculates the weight of node  $j$  as

$$W = \alpha P_{e_j} + \gamma P_{b_j} \quad (1)$$

Where,  $\alpha$  and  $\gamma \in (0,1)$ .  $P_{e_j}$  is the probability that node  $j$  is having sufficient energy remaining to forward the message and  $P_{b_j}$  is the probability that the node  $j$  is having buffer space available to store the message. The node  $j$  is selected as a relay only if the weight of node  $j$  ( $W_j$ ), is greater than a threshold  $T$ . If  $j$  is selected as a relay, a copy of the message is created and forwarded to it.

$$P_e = \frac{1}{1 + e^{-(EL - T_e)}} \quad (2)$$

$$P_b = \frac{1}{1 + e^{-(BS - T_b)}} \quad (3)$$

Where  $EL$  is the remaining energy level of a node and  $BS$  is its available buffer space. The above two probabilities can be defined using the following characteristic functions in the  $(0,1)$  probability space. These functions serve as decision functions in message forwarding.

$$\Psi_{P_e} = \begin{cases} 1, & \text{if } EL > T_e \\ 0, & \text{otherwise} \end{cases}$$

$$\Psi_{P_b} = \begin{cases} 1, & \text{if } BS > T_b \\ 0, & \text{otherwise} \end{cases}$$

where  $T_e$  and  $T_b$  are the threshold values of energy level and buffer space.

#### 4.1 Mathematical modeling of EBR

In this section, we will analytically evaluate the proposed EBR protocol. We formulate the opportunistic message forwarding as fluid-flow Markov-chain process. We have calculated the rate of message propagation (infection) among nodes and optimized the delivery ratio as well as average delivery delay.

Following the epidemic modelling as in [19], let  $I(t)$  be the set of infected nodes (which are having copies of message  $m$ , at time  $t$  and  $\beta$  is the inter-contact time between nodes.

$$\frac{dI(t)}{dt} = \beta I(t)(N - I(t)) \quad (4)$$

Similarly, let  $C(t)$  be the cumulative density function, of message  $m$ , will be delivered in time  $t$

$$\frac{dC(t)}{dt} = \beta I(t)(1 - C(t)) \quad (5)$$

Solving equations 4 and 5 with initial condition  $I(0) = 1$  and  $C(0) = 0$ , gives

$$I(t) = \frac{N}{1 + (N - 1)e^{-\beta N t}} \quad (6)$$

$$C(t) = 1 - \frac{N}{(N - 1) + e^{\beta N t}} \quad (7)$$

Let,  $P_e$  be the probability that a node is having enough energy available for forwarding the message  $m$  ( $EL_i > T_e$ ). Similarly,  $P_b$  is the probability that a node is having sufficient buffer space available ( $BS_i > T_b$ ).

Considering a node's capability of message forwarding based on its remaining energy level as well as buffer space, equations, 4 and 5 can be written as

$$\frac{dI(t)}{dt} = \beta I(t)(N - I(t)) \times \alpha P_e \times \gamma P_b \quad (8)$$

$$\frac{dC(t)}{dt} = \beta I(t)(1 - C(t)) \times \alpha P_e \times \gamma P_b \quad (9)$$

Solving equations 8 and 9 with initial conditions  $I(0) = 1$  and  $C(0) = 0$ , gives

$$I(t) = \frac{N}{1 + (N - 1)e^{-\alpha P_e \gamma P_b \beta N t}} \quad (10)$$

$$C(t) = 1 - \left( \frac{N}{(N - 1) + e^{\alpha P_e \gamma P_b \beta N t}} \right)^{\frac{1}{\alpha P_e \gamma P_b}} \quad (11)$$

The delivery ratio can be calculated as

$$DR(t) = \frac{I(t)}{N} = \frac{1}{1 + (N - 1)e^{-\alpha P_e \gamma P_b \beta N t}} \quad (12)$$

Similarly, average delivery delay can be expressed as

$$DD(t) = \int_0^\infty (1 - C(t)) dt = \ln \frac{N}{\alpha P_e \gamma P_b (\beta N - 1)} \quad (13)$$

## 5. PERFORMANCE EVALUATION

The proposed EBR protocol has been extensively tested using both simulations and real prototype smart vehicular test-bed. In this section, we will discuss the performance metrics, simulation settings and results of the simulation.

Table 1: Simulation settings

Parameter	Value
Simulation time	43200s=12h
Transmission Speed of nodes	250 Kbps
Number of nodes	125
Time To Live (TTL)	300 minutes
Message Creation Interval	25-30 seconds
Message Size	500-1024 KB
Wait Time	10-30 seconds
Device buffer	2-8 MB
Initial energy	4800 mAh
Energy expenditure by scanning	0.92 mAh
Energy spent for message transmission	0.08 mAh
Energy spent for message reception	0.08 mAh

#### 5.1 Performance metrics

We have used delivery ratio, delivery delay and energy dissipation as the performance metrics for evaluating the routing performance of EBR. Delivery ratio is defined as the total number of messages delivered to total number of messages generated for routing. Achieving higher delivery ratio is the main objective of any opportunistic routing protocol. The metric delivery delay is defined as the average time taken to deliver all the messages from source to destination across the network. So the objective of any opportunistic routing protocol is to achieve lower delivery delay. For measuring the energy efficiency, we have used the metric energy dissipation, which is defined as the amount of energy dissipated per unit message delivered. Energy dissipation is calculated as the ratio of total energy consumption in the network to total number of messages delivered. Low energy dissipation implies higher energy efficiency of the opportunistic routing protocol.

#### 5.2 Simulation Setup

We have simulated the scenario of opportunistic STM using Opportunistic Network Environment (ONE) simulator. For simulation using synthetic mobility traces, we have used Shortest Path Map-Based Movement (SPMBM) mobility model. The SPBM mobility model use Helsinki town area map with 125 nodes moving randomly for 12 hours. Since, the total simulation time lasted for 12 hours, a total of 1440 messages were generated during the whole experiment. We have set the initial energy level of each node as 4800 mAh. The transmission and reception of message consumes 0.08 mAh for each node. We consider that there is no random recharge for the nodes and interval of energy recharge is quite high, so that node's energy level is decreasing with message forwarding. The simulation settings are shown in Table 1. We have compared the routing performance of our protocol with Epidemic routing [20] with energy-constraints, n-epidemic [14] with  $n$  as 4 and EAER [15]. We have conducted several experiments with different values of  $\alpha$  and  $\gamma \in (0,1)$  and finally set the value of  $\alpha = 0.7$  and  $\gamma = 0.3$ , which achieves the best performance.

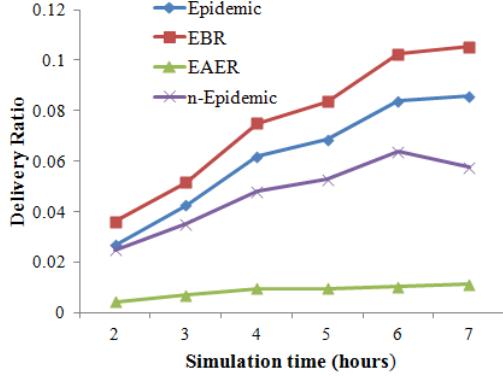


Figure 3: Comparison of delivery ratio

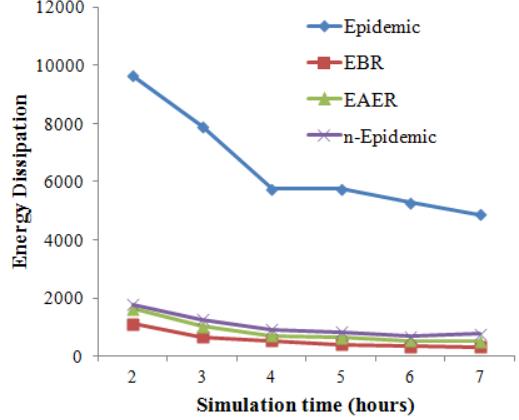


Figure 5: Comparison of energy dissipation

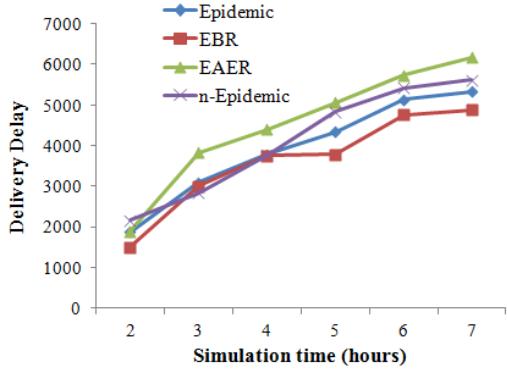


Figure 4: Comparison of delivery delay

### 5.3 Simulation Results

We vary the simulation time from 2 to 7 hours for analyzing the performance metrics delivery ratio, delivery delay and energy dissipation. The results of simulation are discussed below.

Fig.3 shows the comparison of delivery ratio of Epidemic routing, n-Epidemic and EAER varying the simulation time. From the figure, we can observe that the delivery ratio of the proposed EBR protocol outperforms Epidemic routing, n-Epidemic and EAER. The reason is that, in EBR, the nodes with sufficient energy and buffer space are only involved in the message forwarding.

Fig.4 shows a comparison of delivery delay of the routing protocols. Initially the delivery delay of EBR is as par with other routing protocols and as simulation time progress, EBR achieves lowest delivery delay compared to other protocols. This is because, although the number of message forwarding is reduced in EBR, the selected relay nodes are having sufficient energy and buffer space, so that the messages are guaranteed to be delivered to the destination.

Comparing the energy dissipation, as in Fig.5, as simulation progress, although total energy consumption is high, the number of delivered messages also increases, which in turn

reduces the energy dissipation. Compared to other protocols, Epidemic routing is having highest energy dissipation due its flooding nature of message forwarding. Form the figure, it can be observed that EBR is having lowest energy dissipation, which indicates its energy efficiency.

## 6. PROTOTYPE TEST-BED SYSTEM

We have analyzed the performance of the proposed EBR protocol in a smart vehicular test-bed system. The smart vehicular test-bed consists of vehicles equipped with sensors which help them sense the surrounding environment and communicate among each other. We have attached camera sensors to the vehicles, which will detect the motion of other vehicles and prevents the collision of vehicles. We have also used a vehicle tracking system as part of the smart vehicular test-bed, which will help the vehicles to track their own location coordinates. The vehicles communicate with each other opportunistically and share the information, which will help them to regulate their speed. We have evaluated the performance of EBR in real prototype test-bed using the same performance metrics used for simulation (Refer to section 5.1).



Figure 6: Smart vehicular test-bed prototype

### 6.1 Experimental Setup

The proposed smart vehicular test-bed consists of three TRK-MPC5604B free-scale smart motor cars (Fig.6) randomly moving around a long corridor of size  $40 \times 5 m^2$ . The hardware used in the smart vehicular test-bed is shown in

<b>TRK-MPC5604B</b>		The Freescale TRK-MPC5604B micro-controller board is used for controlling the servo motor using microcontroller programming.
<b>Motor Drive Board</b>		The motor drive board also called as power stage board is used for controlling motor based on receiving signal from the micro-controller.
<b>Arduino Uno</b>		Arduino Uno is ATmega328 based micro-controller board where ATmega328 has 32KB memory, from which 0.5KB is used for boot loader installation.
<b>XBee Series 2</b>		The XBee series 2 module is used for wireless communication.
<b>Arduino Wireless SD shield</b>		This shield act as an interface to attach XBee module to the Arduino board.
<b>XBee Explorer</b>		XBee explore is used for configuration of XBee firmware and parameters.

**Figure 7: Hardware used for smart vehicular communication**

Fig.7. The cars will communicate with each other using Xbee module, based on IEEE 802.15.4 protocol. The region is long enough so that the cars frequently move out of each other's communication range and thus get disconnected from each other. In the situations of disconnections, the cars make use of opportunistic communication, thereby storing the messages and carry until they find appropriate relay node or the destination. The position and movements of the cars are controlled using the motor control board attached to the car. We have mounted Arduino micro-controller chip on the car and the sensors attached to the Arduino board, helps sense the movement of the other cars. EBR routing protocol is programmed using ArduinoIDE and uploaded to the Arduino board mounted on the car. The SD card attached to the Arduino wireless SD shield is used to store the messages. We have implemented EBR protocol on Arduino board, which is then mounted on each car.

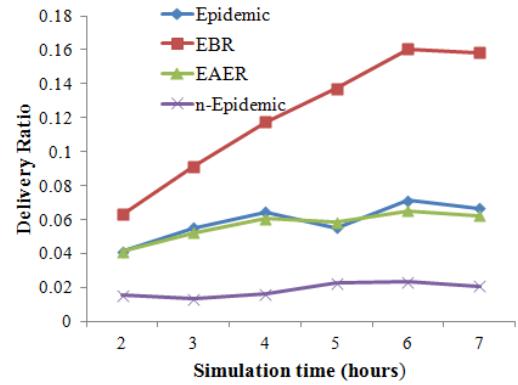
We have implemented Vehicle Tracking System (VTS) on the smart vehicular test-bed (Fig.8) described earlier. The VTS consists of a GPS module and AT89S52 micro-controller. The AT89S52 micro-controller is used for interfacing between the hardware peripherals. The GPS modem will continuously send the location coordinates of the vehicle to the other vehicles through car-mounted communication system. When the cars get disconnected the messages are forwarded using opportunistic encounters.

## 6.2 Results

Fig.9 depicts the comparison of delivery ratio of the EBR with Epidemic, n-Epidemic and EAER routing protocol us-



**Figure 8: Vehicle tracking system as part of real test-bed**



**Figure 9: Delivery ratio using real test-bed analysis**

ing the prototype test-bed. From the figure, we can observe that the delivery ratio of EBR is higher than Epidemic routing and other protocols. The reason is that in the case of EBR, the messages are forwarded to nodes having sufficient energy and buffer space, so that, there is a higher chance of delivering the message to the destination compared to other protocols. Similarly, Fig.10 shows the comparison of the delivery delay using the prototype real test-bed. From the figure, it can be observed that the delivery delay of EBR is less compared to other protocols. Similarly, compared to other protocols, EBR achieves lowest energy dissipation (Fig.11), which is an indication of higher energy efficiency. This is because, the energy level of the nodes are efficiently utilized by allowing only nodes having sufficient energy to forward the messages further in the network.

In summary, as a proof-of-concept we implemented EBR protocol in a smart vehicular environment. In the future, we will enhance our vehicular test-bed with more cars, for conducting additional experiments.

## 7. CONCLUSION

Developing applications to provide smart city solutions is an emerging area of research as well as lucrative domain for

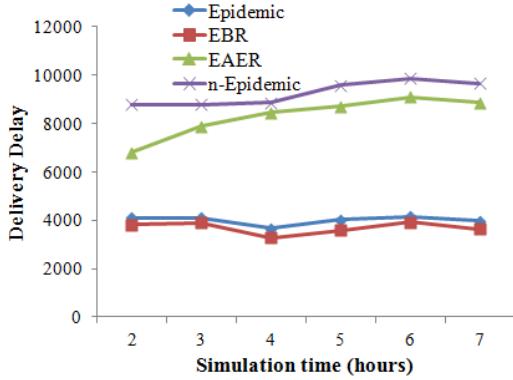


Figure 10: Delivery Delay using real test-bed analysis

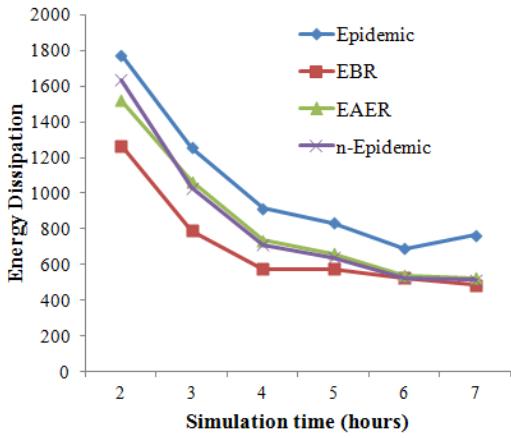


Figure 11: Energy Efficiency using real test-bed analysis

business. STM is one of such a futuristic application, where the sensor nodes attached to the vehicles help make decisions based on the sensed data. Since sensors used in such a system are energy-constrained, and the message forwarding is an energy consuming process, designing an energy-efficient message forwarding in an opportunistic STM is a complicated task. In this paper, we have developed an energy-efficient and buffer-aware routing protocol for an opportunistic STM. Extensive simulations as well as real test-bed experiments show that the proposed EBR protocol achieves higher energy efficiency, higher delivery ratio and lower delivery delay.

## 8. ACKNOWLEDGMENTS

This work is partially supported by the Alexander von Humboldt Foundation through the post-doctoral research fellowship of Dr. Vaskar Raychoudhury. The authors would like to thank Mr. Deepak for his help in this work.

## 9. REFERENCES

- [1] A. Z. Kausar, A. W. Reza, M. U. Saleh, and H. Ramiah, "Energizing wireless sensor networks by energy harvesting systems: Scopes, challenges and approaches," *Renewable and Sustainable Energy Reviews*, vol. 38, pp. 973–989, 2014.
- [2] W. Huang, H. Chen, Y. Li, and B. Vucetic, "On the performance of multi-antenna wireless-powered communications with energy beamforming," 2015.
- [3] A. Berger, L. B. Hormann, C. Leitner, S. B. Oswald, P. Priller, and A. Springer, "Sustainable energy harvesting for robust wireless sensor networks in industrial applications," in *Sensors Applications Symposium (SAS), 2015 IEEE*. IEEE, 2015, pp. 1–6.
- [4] L. Pelusi, A. Passarella, and M. Conti, "Opportunistic networking: data forwarding in disconnected mobile ad hoc networks," *Communications Magazine, IEEE*, vol. 44, no. 11, pp. 134–141, 2006.
- [5] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: Social-based forwarding in delay-tolerant networks," *Mobile Computing, IEEE Transactions on*, vol. 10, no. 11, pp. 1576–1589, 2011.
- [6] T. Abdelkader, K. Naik, A. Nayak, N. Goel, and V. Srivastava, "Sgbr: A routing protocol for delay tolerant networks using social grouping," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, no. 12, pp. 2472–2481, 2013.
- [7] Q. Li, W. Gao, S. Zhu, and G. Cao, "A routing protocol for socially selfish delay tolerant networks," *Ad Hoc Networks*, vol. 10, no. 8, pp. 1619–1632, 2012.
- [8] E. Talipov, Y. Chon, and H. Cha, "Content sharing over smartphone-based delay-tolerant networks," *Mobile Computing, IEEE Transactions on*, vol. 12, no. 3, pp. 581–595, 2013.
- [9] E. Bulut and B. K. Szymanski, "Exploiting friendship relations for efficient routing in mobile social networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, no. 12, pp. 2254–2265, 2012.
- [10] C. Sabin, V. Raychoudhury, G. Marfia, and A. Singla, "A survey of routing and data dissemination in delay tolerant networks," *Journal of Network and Computer Applications*, 2016.
- [11] D. R. Silva, A. Costa, and J. Macedo, "Energy impact analysis on dtn routing protocols," 2012.
- [12] A. Keränen, J. Ott, and T. Kärkkäinen, "The one simulator for dtn protocol evaluation," in *Proceedings of the 2nd international conference on simulation tools and techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, p. 55.
- [13] Y. Li, Y. Jiang, D. Jin, L. Su, L. Zeng, and D. Wu, "Energy-efficient optimal opportunistic forwarding for delay-tolerant networks," *Vehicular Technology, IEEE Transactions on*, vol. 59, no. 9, pp. 4500–4512, 2010.
- [14] X. Lu and P. Hui, "An energy-efficient n-epidemic routing protocol for delay tolerant networks," in *Networking, Architecture and Storage (NAS), 2010 IEEE Fifth International Conference on*. IEEE, 2010, pp. 341–347.
- [15] F. De Rango, S. Amelio, and P. Fazio, "Enhancements of epidemic routing in delay tolerant networks from an energy perspective," in *Wireless communications and*

- mobile computing conference (IWCMC), 2013 9th international.* IEEE, 2013, pp. 731–735.
- [16] M. Khouzani, S. Eshghi, S. Sarkar, N. B. Shroff, and S. S. Venkatesh, “Optimal energy-aware epidemic routing in dtns,” in *Proceedings of the thirteenth ACM international symposium on Mobile Ad Hoc Networking and Computing*. ACM, 2012, pp. 175–182.
  - [17] Y. K. Yao, W. X. Zheng, and Z. Ren, “An energy-efficient routing algorithm for disruption tolerant networks,” in *Advanced Materials Research*, vol. 756. Trans Tech Publ, 2013, pp. 3754–3759.
  - [18] W. Feng and S. Li, “Energy efficient terminal-discovering in mobile delay tolerant ad-hoc networks,” in *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2013 International Conference on*. IEEE, 2013, pp. 465–470.
  - [19] X. Zhang, G. Neglia, J. Kurose, and D. Towsley, “Performance modeling of epidemic routing,” *Computer Networks*, vol. 51, no. 10, pp. 2867–2891, 2007.
  - [20] A. Vahdat, D. Becker *et al.*, “Epidemic routing for partially connected ad hoc networks,” Technical Report CS-200006, Duke University, Tech. Rep., 2000.

# Total Order Reliable Convergecast in WBAN

Gewu Bu<sup>\*</sup>

Université Pierre et Marie Curie, LIP6, CNRS  
UMR 7606  
4, place Jussieu, Paris, France  
gewu.bu@lip6.fr

Maria Potop-Butucaru

Université Pierre et Marie Curie, LIP6, CNRS  
UMR 7606  
4, place Jussieu, Paris, France  
maria.potop-butucaru@lip6.fr

## ABSTRACT

This paper is the first extensive work on total order reliable convergecast in multi-hop Wireless Body Area Networks (WBAN). Convergecast is a many-to-one cooperative scheme where each node of the network transmits data towards the same sink. Our contribution is threefold. First, we stress existing WBAN convergecast strategies with respect to their capacity to be *reliable* and to ensure the *total order* delivery at sink. That is, packets sent in a specific order should be received in the same order by the sink. When stressed with transmission rates up to 500 packets per second the performances of these strategies decrease dramatically (more than 90% of packets lost). Secondly, we propose a new posture-centric model for WBAN. This model offers a good characterization of the path availability which is further used to fine tune the retransmission rate thresholds. Third, based on our model we propose a *new mechanism for reliability* and a *new converge-cast strategy* that outperforms WBAN dedicated strategies but also strategies adapted from DTN and WSN areas. Our extensive performance evaluations use essential parameters for WBAN: packet loss, total order reliability (messages sent in a specific order should be delivered in that specific order) and various human body postures. In particular, our strategy ensures zero packet order inversions for various transmission rates and mobility postures. Interestingly, our strategy respects this property without the need of additional energy-guzzler mechanisms.

## Keywords

Wireless Body Area Networks, Total Order Reliable Convergecast, Networks Modelization

## 1. INTRODUCTION

Wireless Body Area Networks (WBAN) is a cross-area between Wireless Sensor Networks (WSN) and Delay Tolerant

<sup>\*</sup>This work has been supported by the SMARTBAN project, Labex SMART

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

ICDCN '17, January 04-07, 2017, Hyderabad, India

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007761>

Networks (DTN) with as main objective the collection of physiological parameters from sensors deployed on a human body that undergo the human mobility. Designing efficient protocols for this area is a challenging task. One of the particularities of the WBAN area are the specific rates of the physiological parameters monitoring. In [1] the authors advocate that in the current applications of WBAN, the transmission rate is less than 10 packets per second. However, these rates may drastically increase in the future. Therefore, in order to meet the requirements of medical monitoring, WBAN is needed to withstand a continuous packets flows. In some critical applications such as monitoring patients during a surgery or post-surgery the flows carry vital alerts. Missing some of them or wrongly interpret the flow of data due to packets inversions may have a tremendous impact. In this context designing efficient total order reliable protocols is crucial for saving human lives.

In this paper we focus the *convergecast* communication primitive since this is one of the main building block in WBAN. Convergecast allows nodes to transmit data towards a sink. Convergecast has the total order property if packets are delivered at the sink in the same order as they have been transmitted.

Recently, [2] surveys the existing work on convergecast on various areas including Delay Tolerant Networks, DTN and Wireless Sensor Networks, WSN. They argue that most of the existing strategies are not directly implementable in WBANs due to their needs in computing capacities, memory or their energy consumption. Finally, building on top of [3, 4, 5, 6, 7], they propose and evaluate three classes of convergecast strategies taylored for WABN: 1) Multi-Paths based strategies, 2) Attenuation-based strategies and 3) Gossip-based strategies. Their evaluation does not target the total order delivery property of these strategies neither their resistance to increasing flows of messages. It merely focus on the resilience to the human body mobility and energy consumption. They measure the following parameters: percentage of received messages (under the hypothesis that each node sends a single message in each run), end-to-end delay and the number of transmission/receptions since this is a good indicator of the energy consumption.

In terms of *reliability*, to the best of our knowledge there are three main reliability mechanisms used in WBAN. The first mechanism computes and presesets a static or dynamic overlay path for collecting data. Strategies proposed in [8, 9] and attenuation-based strategies in [2] choose a reliable leader or a set of relay nodes as the next hop(s) to help in forwarding the packets. Strategies proposed in [10, 11,

12, 13, 14, 15, 16, 17] care and learn in the entire network paths by sending BEACON or HELLO message in order to construct a path from every source to the sink. These paths are then dynamically updated during the packets sending. The Multi-Paths based strategies in [2] use a specific static path to collect data.

The second mechanism use special retransmission techniques at MAC level [18] or a TCP-like reliable flow control mechanism at Transport level [19]. However, they are more like Cross-Layer strategies [20, 21]. In this work we specifically address only the network level strategies.

The third mechanism uses either compressive sensing (CS) technology to reconstruct missing packets [22] conservation central network CCN-based model [23]. This third mechanism is particularly energy-guzzler.

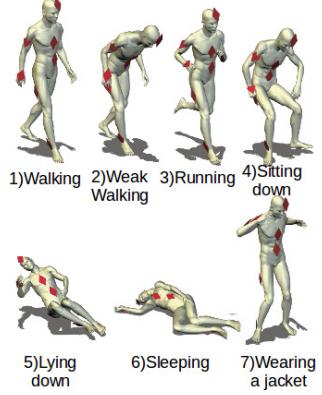
In terms of *total order* (packets delivery that respect the sending order), to the best of our knowledge no extensive study has been conducted so far in WBAN.

### Our Contribution.

Our work on multi-hop WBAN was motivated by the recent findings in [24] where it is proven that multi-hop communication has better performances face to human body mobility than classical one hop communication. Our work is the first to extensively study the total order reliability of convergecast in multi-hop WBAN. Our contribution is *threefold*. *First* we simulate and analyse thirteen representative strategies for convergecast some of them tailored for WBAN others borrowed from the sensor network or DTN literature. We stress them with various packets rate up to 500 packets per second. We evaluate the throughput as well as their capability to totally order delivered packets (packets are delivered at the sink in the same order as they have been transmitted). We performed our evaluations with OMNeT++ simulator that we enriched with realistic human body mobility and channel models issued from the recent research on biomedical and health informatics [24]. Our simulations show that the throughput is inversely proportional with the increase of transmission rate. The packets loss goes to 90%. This first contribution is of independent interest since it lays the basis for designing efficient convergecast strategies in multi-hop WBAN. Our *second contribution* is a *new model for WBAN* and based on this new model we propose a *new mechanism* for increasing the reliability of convergecast. Interestingly, equipped with this new mechanism convergecast strategies improve their performances both in terms of throughput and total order reliability for a packet rate that goes as far as 20 packets per second. Note that this threshold is twice bigger than the maximum packet rate of actual WBAN applications [1]. However, this mechanism does not help in avoiding sequence inversions. Our *third contribution* is a new convergecast strategy that ensures 100% total order reliability.

### Roadmap.

In Section 2 we simulate and analyse thirteen different strategies: four Multi-Paths based strategies [2] as examples of the presetting reliable overlay paths; four Attenuation-based strategies [2] as examples of reliable choice of the next-hop; the Collecting Tree Protocol (CTP) [10] and the Opportunistic Routing (ORW) [11] as examples of entire network reliable strategies. Furthermore, we analyse three Gossip-based strategies [2] as example of flooding-based strategies.



**Figure 1: Seven postures for different human mobility**

In Section 3 we present our new model for WBAN. In Section 4 we introduce our new reliability mechanism. Finally, in Section 5 we introduce our new convergecast strategy that out-performs existing ones in terms of total order reliability.

## 2. ANALYSIS OF EXISTING CONVERGECAST STRATEGIES

In this section, we stress strategies discuss in [2] plus the strategies CTP [10] and ORW [11]. More specifically, we analyse their throughput when the transmission rate goes as far as 500 packets per second. Our simulations prove that their performances (without exception) proportionally decrease with the increase of the transmission rate.

In the sequel we present briefly the channel model we used, then the simulation environment. Furthermore, we briefly present each of the strategies we simulated and finally the simulation results.

### Channel Model and Simulation Environment.

In this work we use the channel mobility model proposed in [24]. In [24] the authors proposed a simulation-base channel model based on mobility data set for WBAN issued from experiments with a network composed of seven sensors distributed on the body as follows : 0)navel, 1)chest, 2)head, 3)upper arm, 4)ankle, 5)thigh and 6)wrist. Using a software-simulated-human-body model to replace the real person, the authors measure the mean and the standard deviation of the channel attenuation between every two nodes in seven different postures: 1)Walking, 2)Walking weakly, 3)Running, 4)Sitting down, 5)Lying down, 6)Sleeping and 7)Wearing a jacket, respectively (see Figure 1).

Furthermore, the authors of [24] advocate that multi-hop communication achieves better performances than one-hop star topology considered so far in WBAN. Based on this data set, [25] proposed a new channel-mobility: for every packet sent from a source, a random attenuation will be added. If the signal strength after adding the attenuation is smaller than the sensibility at the receiver, then the packet will be dropped.

In this paper, we use the same simulation environment as in [25] (IEEE 802.15.4) with a communication frequency of 2.45 GHz.

The transmission power and the sensibility of the radio

module of nodes are set to -60dBm and -100dBm respectively.

## 2.1 Description of analyzed strategies

Convergecast is a many-to-one cooperative scheme where each node of the network transmits data towards the same sink. In [2] the authors survey and classify existing convergecast strategies for WBAN. We simulated these strategies altogether with CTP [10] and ORW [11] on top of a network composed of seven nodes using a channel and body mobility model as described in the previous section.

In the sequel we present briefly the simulated strategies followed by the detailed presentation of the simulation results. As explained previously, we consider multi-hop communication and the human mobility model presented in [24] Figure 1.

*Multi-Paths based strategies* : these strategies are based on predetermined paths and use this pre-set overlay as a reliability mechanism.

- APAP: Each node sends or forwards packets to all its parents until packets reach the sink.
- APPP: Each source node sends its packets to all its parents and parents randomly forward packets to one of their parents.
- PPAP: Each source node randomly sends its packets to one of its parents and parents forward packets to all their parents.
- PPPP: Each node randomly sends or forwards packets to one of its parents.

*Attenuation-based strategies* : these strategies are based on the negotiation of channel attenuation. When a source has packets to send, it broadcasts firstly a Request to ask an estimated attenuation from the receiver of this Request to the sink then the receiver of this Request will send back a Reply with the required attenuation value. The source receiving Replies will chose a next hop among replying nodes and sends data packets to them; if no Reply has been received for a while, the source will re-send a Request.

- MinAtt: Each source chooses as next hop the nodes with the best (smallest) attenuation value among the responders.
- BothMinAtt: Each source chooses as next hop two nodes who give the two best (smallest) attenuation values among all the responders.
- CloseToMe: The source requires not only the attenuation value to the sink but also the attenuation to the source and chooses two nodes: the ones that give the two best (smallest) attenuation valuers, according to who has a smaller attenuation value to the source.
- RandAtt: The source randomly chooses one responder as the next hop.

*Dynamic Path Strategies* : these strategies construct and update an overlay route over time.

- CTP: All sources periodically broadcast BEACON messages for constructing and updating a network-cost metric, according to which each node chooses his next hop to the sink.

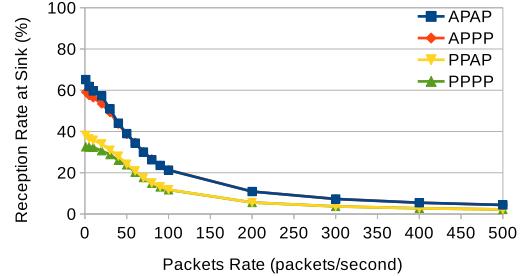


Figure 2: Reception Rate of Multi-Paths based strategies

- ORW: Nodes get their parents in an initial phase where PROBING and REPROBING messages are exchanged. After this initial phase, nodes update the network-cost metric by computing data packets's receiving rate on each node. Sources send packets to a set of relay nodes.

*Gossip-based strategies* : these strategies are based on packets flooding.

- FloodToSink: Nodes broadcast their own packets and the received packets to their neighbours until packets reach the sink or the limitation of TTL reached.
- ProbaCvg: Nodes broadcast their own packets and the packets received to their neighbours with a probability until packets reach the sink or the limitation of TTL reached. Each time of broadcast will halve the forwarding probability.
- PrunedCvg: Nodes send their own packets and the packets received uni-cast to a random node as next hop.

## 2.2 Simulation Results

Due to the lack of space, in this section we present solely the results for the *Walking* posture. We measure the percentage of received packets by the sink when the packets rate varies from 1 packet to 500 packets per second. Furthermore, we address the total order property.

### 2.2.1 Throughput

In *Multi-Paths based strategies* (Figure 2), the reception rate decreases fast with the increase of the transmission rate. APAP and APPP have better reception rate than PPAP and PPPP. The reason is that APAP and APPP strategies allow source nodes to send packets to both their parents to increase (in particular for sources who are farthest from the sink) the probability of their packets reception. Moreover, APAP is better than APPP and PPAP is better than PPPP, because in APAP and PPAP nodes have more chances to see their packets forwarded by their parents than in APPP and PPPP, respectively.

*Attenuation-based strategies* (Figure 3) have almost the same reception rate tendency and it decreases fast when transmission rate is superior to 50 packets par second. The reason is that the Request/Reply mechanism is inefficient and heavy in the intermittent environment of WBAN. The missing Reply for corresponding Request prevents sources from sending packets to next hop(s). Also, these strategies

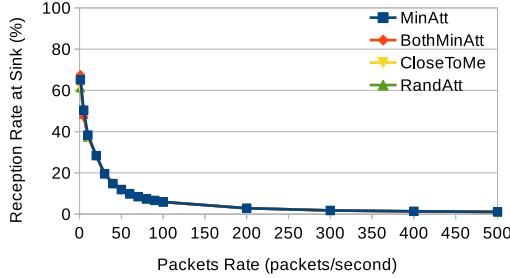


Figure 3: Reception Rate of Attenuation based strategies

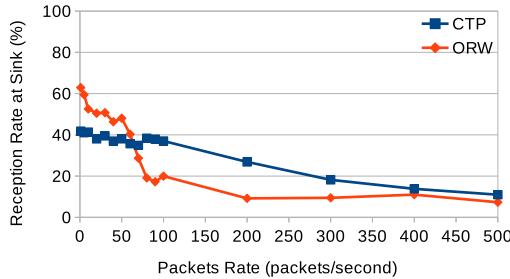


Figure 4: Reception Rate of Dynamic Path based strategies

have important packets loss due to the over-buff of the waiting queues.

*Dynamic Path strategies* simulations are presented in Figure 4. ORW is better than CTP for transmission rates inferior to 75 packets par second. After that, ORW decrease faster than CTP. The reason is that CTP dynamically chooses the best next hop, but the in-band BEACON message affect the reception rate of useful packets. ORW chooses a set of next hops which increase the reception rate obviously when the transmission rate is not high enough. Furthermore, the increase of the transmission rate causes more and more collisions which yields to a fast decrease of the reception rate.

In the set of *Gossip-based strategies* (Figure 5), FloodToSink shows a good reception rate at the beginning, but from the same reason as ORW, the reception rate decreases fast. ProbaCvg and PrunedCvg are worse than FloodToSink at the beginning, but they decrease not as fast as FloodToSink due to the fewer packets flooding into the network in ProbaCvg and PrunedCvg.

The percentage of delivered messages at sink is relatively low and the reception rate decreases rapidly with the increase of the traffic rate.

The reason for the low reception rate is the intermittent network connection due to the important attenuation of the channel-mobility model. Reaching the limitations of network throughput is the reason for the rapid decrease of reception rate with the growing of data rate. The network reaches its limitations due to more and more collisions, interferences, packet and over-buff errors.

### 2.2.2 Total order Analysis

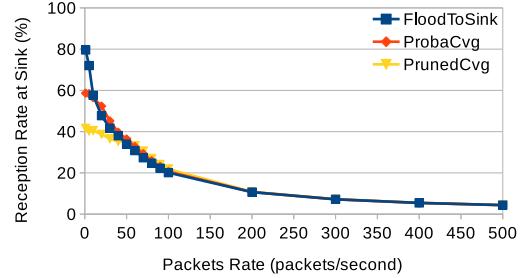


Figure 5: Reception Rate of Gossip-based strategies

In the section we analyse the ability of the studied strategies to deliver packets at the sink respecting the total order property: packets from the same source should be received at the sink in the same order as they have been sent. From our simulations none of the 13th strategies has a zero wrong-sequence packets.

The main reason for this wrong-sequence delivery is the fact that all strategies create virtual multi-paths during packet transmissions.

We performed additional simulations concerning traditional mechanisms such as retransmissions and sequences. Due to lack of space, we only resume our conclusions. The reception rate when using the ACK-based retransmission is worse than when no retransmission is used. When using sequencer mechanisms at the sink level we obtain a end-to-end delay very important or we have to sacrifice the rate reception. The analysis performed in the previous sections and additional simulations focusing the traditional mechanisms (e.g. retransmissions and sequencers) conducted us to propose a new reliability mechanism based on a new channel model and a new convergecast strategy that outperform existing one.

## 3. NEW MODEL

From the mean and the standard deviation for each link in [24], we can thus calculate the Cumulative Distribution Function (CDF) of the random attenuation:  $F(X) = P[x < X]$ .  $F(X)$  is the probability of the random event:  $x$  is smaller than  $X$ , where  $x$  is the random attenuation variable and  $X$  is a threshold. According to our channel-mobility model, links appear and disappear from time to time, due to the channel attenuation. A connexion exists if the transmission power minus the attenuation is bigger than the sensibility and doesn't exist if not.

So if  $X$  is the maximum acceptable attenuation in a link for transmission,  $F(X)$  is the probability that the random attenuation is smaller than the maximum acceptable attenuation. In other words the connexion exists and the transmission may be successful.  $1/F(X)$  is thus the Expected Transmission Count (ETX) in this link for a transmission.

In our case, we use  $-60dBm$  as the initial transmission power,  $-100dBm$  as sensibility. It follows that the maximum acceptable attenuation is  $(-60dBm - (-100dBm)) = 40dB$ . If  $X = 40dB$ , then  $F(40)$  is the probability for a successful transmission and  $1/F(40)$  is the ETX of the transmission in this link. Since we have all the means and standard deviations so we can compute all the ETX for every link.

P1	0	1	2	3	4	5	6
0		1		0.22404		0.42074	
1			0.99865	0.45407		0.0782902	
2				0.14484			
3					0.125932	0.991802	
4					0.274253		
5						0.935133	
6							
P2	0	1	2	3	4	5	6
0		1	0.0172454	0.0715087	0.0145615	0.1666719	
1			0.365112	0.537937	0.0195801	0.441826	
2				0.438505		0.0580416	
3					0.0140774	0.0559463	0.911143
4						0.710743	0.0121191
5							0.203952
6							
P3	0	1	2	3	4	5	6
0		1	0.0324349	0.217161	0.0129645	0.983938	
1			0.996679	0.688255	0.090184	0.248252	
2				0.254061		0.0309741	
3					0.0123552	0.144004	0.961984
4						0.137656	0.0276401
5							0.683416
6							
P4	0	1	2	3	4	5	6
0		1	0.245884	0.388581	0.0938422	0.619569	
1			0.999912	0.797672	0.0189714	0.272997	
2				0.369441		0.104822	
3					0.0508818	0.656713	
4					0.443202	0.017682	
5						0.761193	
6							
P5	0	1	2	3	4	5	6
0		0.999992	0.0611182		0.101297	0.0618189	
1			0.916915	0.209107	0.0784109	0.092332	
2				0.537937	0.032768	0.0738969	
3					0.158655	0.736169	
4					0.433816	0.312214	
5						0.427863	
6							
P6	0	1	2	3	4	5	6
0		0.973211				0.158655	
1			0.151904				
2				0.535259	0.0326943	0.364212	
3					0.149528	0.994261	
4					0.999999		
5							
6							
P7	0	1	2	3	4	5	6
0		0.999895	0.250324		0.0227501	0.0524422	
1			0.764921	0.012699	0.0196703	0.110812	0.0323404
2				0.158655	0.0473185	0.124549	
3						0.968142	
4						0.615666	0.615666
5							0.055088
6							

Figure 6: Links-existence probability for each link in 7 Postures

Figure 6 shows all the connexions probability for each link in different postures. In the sequel we consider only the probabilities greater than 0.01.

#### 4. NEW RELIABILITY MECHANISM

In the following we propose a No-ACK Retransmission Mechanism based on our channel-mobility model. The idea is that each node transmits the same packet expected retransmission count ( $ETX$ ) times without waiting for ACK message. The  $ETX$  value is according to the model described in the previous section. After sending a packet  $ETX$  times, nodes continue to send the next packet following the same mechanism.

Figure 7, Figure 8 and Figure 9 represent the comparative results between original strategies and strategies using our reliability mechanism for Multi-Paths based strategies, Attenuation-based strategies and Gossip-based strategies. Red and Blue curves report the percentage of delivered messages for the original strategies, respectively enhanced

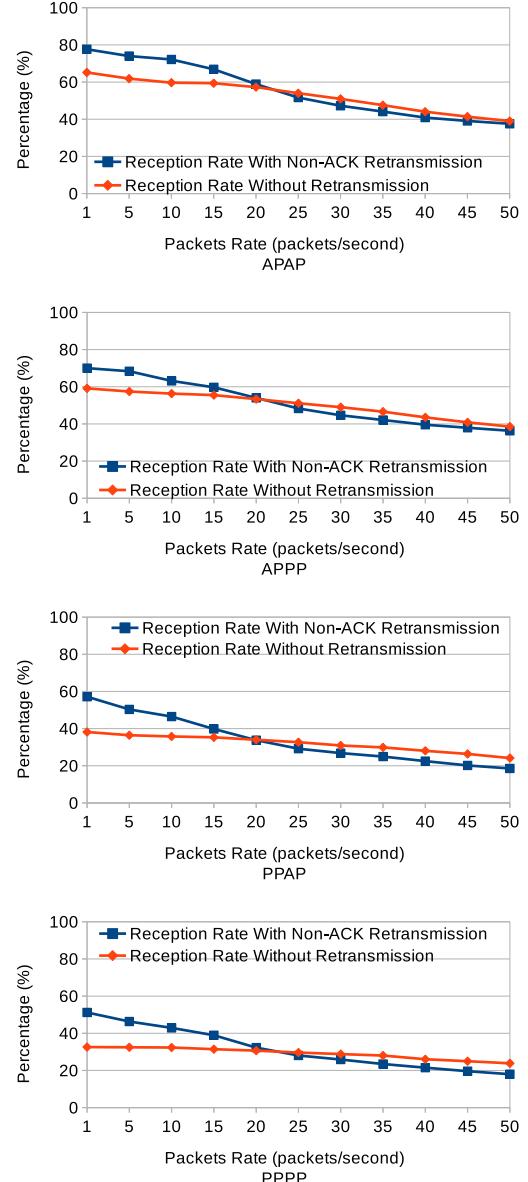


Figure 7: Comparison of using No-ACK Retransmission or not in terms of Reception Rate for Multi-Paths based Strategies

strategies.

Our retransmission mechanism do help to improve the reliability, but with certain limitations:

For Multi-Paths based strategies, our No-ACK mechanism improves the reception rate compared with original strategies up to a transmission rate of 20 packets per second.

For Attenuation-based strategies, our mechanism does not show a great improvement.

For Gossip-based strategies our mechanism makes the reception rate lower than the originals. Note that ProbaCvg strategy will not be well affected by the retransmission mechanism because of its special particularity.

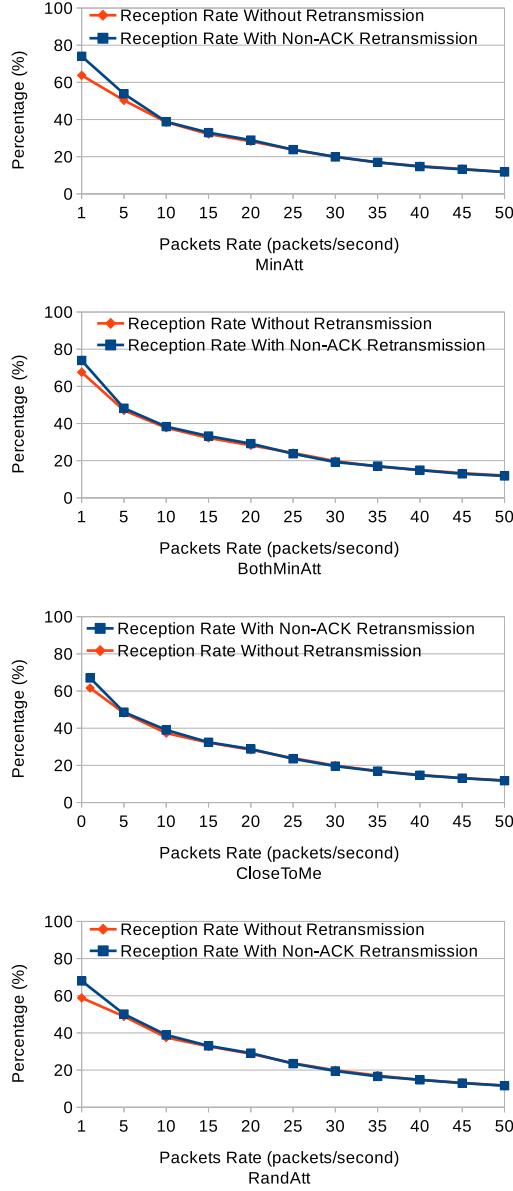


Figure 8: Comparison of using No-ACK Retransmission or not in terms of Reception Rate for Attenuation-based Strategies

## 5. NEW CONVERGECAST STRATEGY

In this section we propose a new convergecast tree based strategy, Probabilist Posture Varying Graph (PPVG). The idea is that each source has only one preselected parent. Parents forward the traffic to their parents until it reaches the sink. For each transmission, we use No-ACK retransmission to ensure the reliability. The preselected path is calculated from links-existence probability (see Figure 6) and ETX of each link. A preselected path is a tree-based multi-hops path (sink as root) for each source node, having the smallest total *ETX* for each source node. We have thus 7 different PPVG Trees for each posture (see Figure 10). Numbers on each link are the *ETX* (trying times of transmission) for each

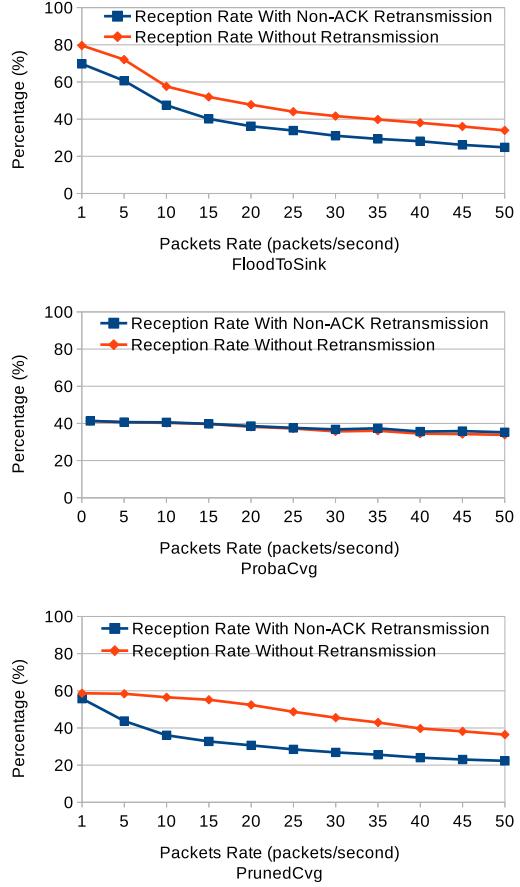


Figure 9: Comparison of using No-ACK Retransmission or not in terms of Reception Rate for Gossip-based Strategies

link; arrows indicate the parents.

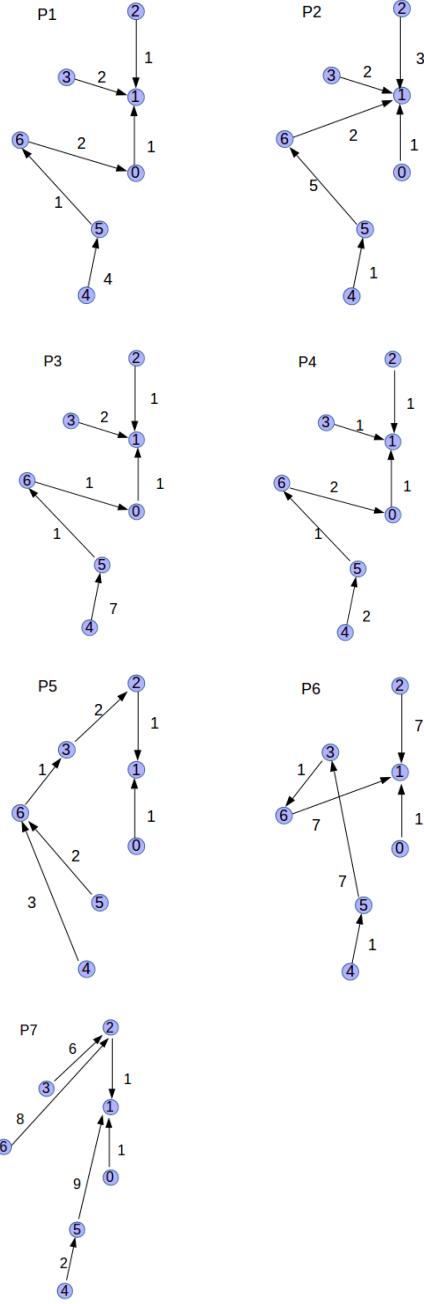
In the sequel we simulate the new PPVG Tree strategy using the same simulation environment as for the previous strategies. The simulation results are shown in Figure 11 for each posture, each of them takes a mean value over 10 simulation runs.

The reception rate of our strategy is stable and slow-decreases except for posture 6 and 7, which presents a clear decrease for packets rate greater than 5 packets per second.

Also, our simulation results show that our strategy preserves the total order (no wrong-sequence) for each posture.

From Figure 12 to Figure 17, we compare all the above described strategies in terms of: 1) Percentage of received messages (Figure 12 and Figure 13) 2) Wrong Sequence rate (Figure 14 and Figure 15) and 3) Number of Transmissions (Figure 16 to Figure 17) at a rate of 10 packets per second (which is the maximum rate of actual WBAN medical applications). All the strategies use the No-ACK Retransmission mechanism. CTP and ORW use the traditional ACK-based Retransmission mechanism. The presented results are the mean values over 10 simulation runs.

Our PPVG Tree strategy has the best Reception Rate in postures 3, 6 and 7; the second best in postures 1 and 2; the third best in postures 4 and 5. Since our PPVG

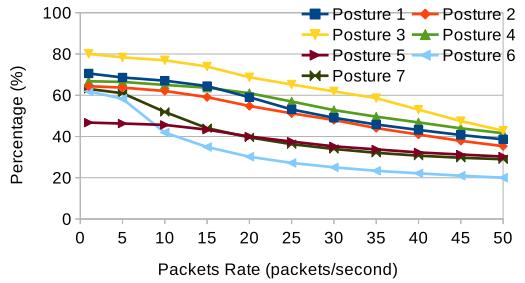


**Figure 10: Different PPVG Trees in different Postures**

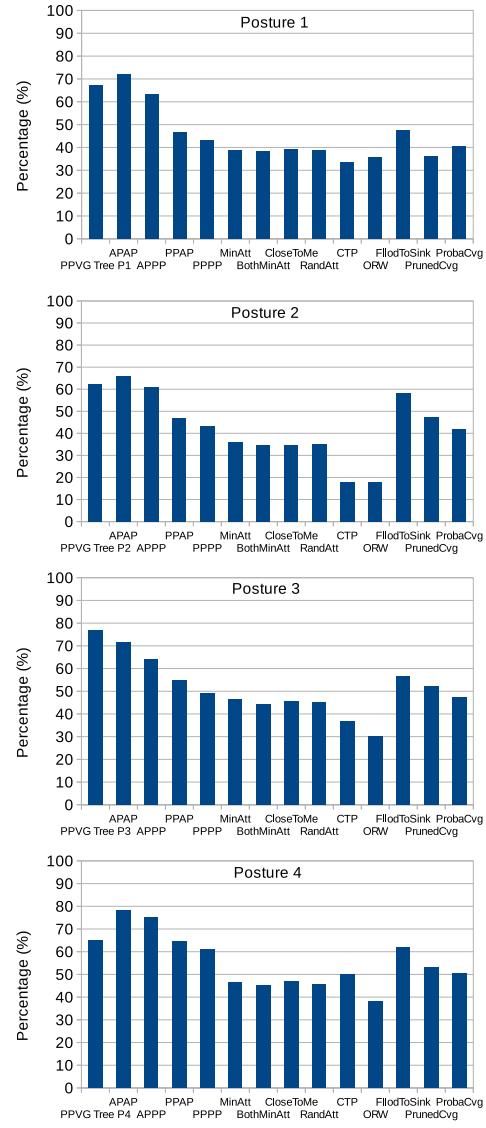
Tree strategy is specified for different postures and doesn't need additional route-updating cost, our strategy gives a reception rate stable and relative high for every postures.

Moreover, our PPVG Tree strategy presents a zero Wrong-Sequence rate for each posture, a good property of total order.

For the number of transmissions, our PPVG Tree strategy always has the smallest transmission number compared with strategies who have comparable Reception Rate with PPVG, like APAP, APPP and FloodToSink.



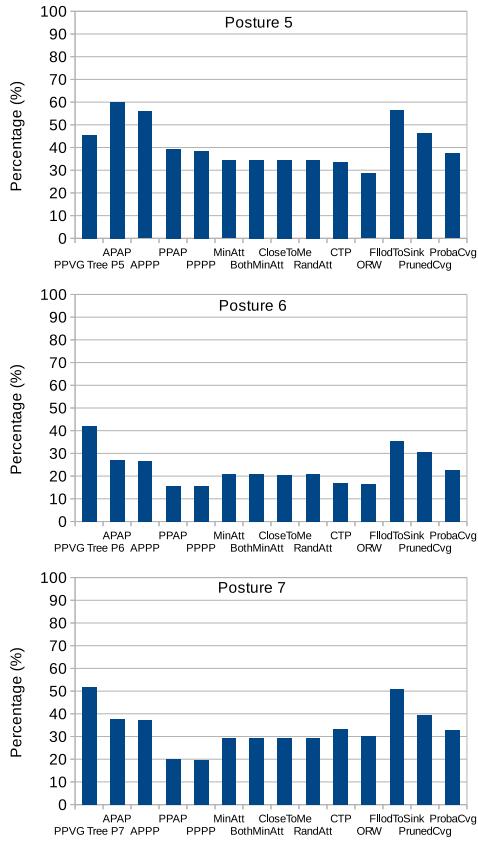
**Figure 11: Reception Rate for PPVG Tree strategy in different Postures**



**Figure 12: Reception Rate comparison among all the strategies in Posture 1-4**

## 6. CONCLUSION

We first stressed existing WBAN convergecast strategies

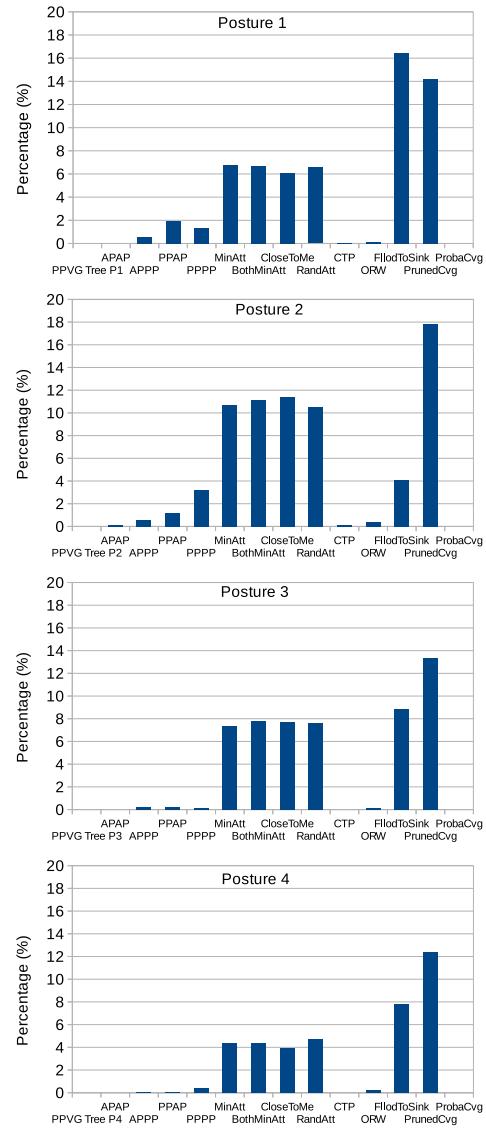


**Figure 13:** Reception Rate comparison among all the strategies in Posture 5-7

with rates of transmissions up to 500 packets per second in order to evaluate their capacity to be reliable and to ensure the total order message delivery. Our findings show that the performances of these strategies decrease dramatically (more than 90% of the packets are dropped). Second, we proposed a new posture-centric model for WBAN and based on this model we proposed a new mechanism for reliability. This mechanisms improves the existing strategies in terms of percentage of delivered messages. We then proposed a new converge-cast strategy that outperforms a broad class of WBAN dedicated strategies but also strategies adapted from DTN and WSN areas. Our extensive performance evaluations target: energy, percentage of message delivery and total order reliability (messages sent in a specific order should be delivered in that specific order). We used various rates of transmission and various human body postures. Our strategy ensures zero packet order inversions. As future work we intend to investigate the fault-tolerance, security and privacy in WBAN.

## 7. REFERENCES

- [1] Jamil Y Khan and Mehmet R Yuce. Wireless body area network (wban) for medical applications. *New Developments in Biomedical Engineering. INTECH*, 2010.
- [2] Wafa Badreddine, Nesrine Khernane, Maria Potop-Butucaru, and Claude Chaudet. Convergecast in wireless body area networks. *Technical report*



**Figure 14:** Packets Inversion Rate comparison among all the strategies in Posture 1-4

- UPMC Sorbonne UniversitÃ's,  
<https://hal.archives-ouvertes.fr/hal-01301773>, 2015.
- [3] Yu-Chee Tseng, Sze-Yao Ni, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. *Wireless networks*, 8(2-3):153–167, 2002.
  - [4] Jitender Grover, Shikha Sharma, and Mohit Sharma. Reliable spin in wireless sensor network: A review. *IOSR Journal of Computer Engineering (IOSR-JCE)*, ISSN, pages 2278–0661.
  - [5] K Karthikeyan and M Kavitha. Comparative analysis of data centric routing protocols for wireless sensor networks. *International Journal of Scientific and Research Publications*, 3(1):1–6, 2013.
  - [6] Kalpana Sharma, Neha Mittal, and Priyanka Rathi. Performance analysis of flooding and spin in wireless sensor networks. *International Journal of Future*

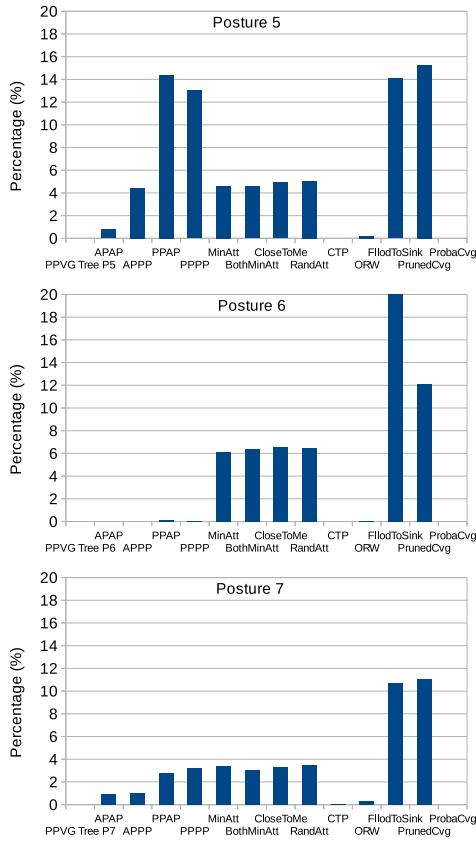


Figure 15: Packets Inversion Rate comparison among all the strategies in Posture 5-7

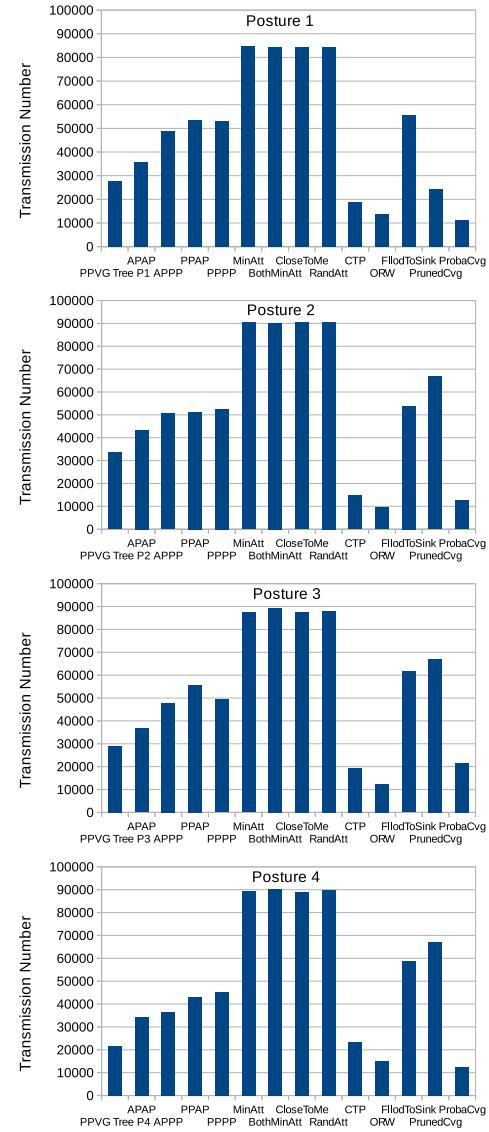


Figure 16: Number of Transmissions comparison among all the strategies in Posture 1-4

- Generation Communication and Networking*, 7(3):25–36, 2014.
- [7] Zeenat Rehena, Sarbani Roy, and Nandini Mukherjee. A modified spin for wireless sensor networks. In *Communication Systems and Networks (COMSNETS), 2011 Third International Conference on*, pages 1–4. IEEE, 2011.
  - [8] BeomSeok Kim, Jinsung Cho, Seokhee Jeon, and Ben Lee. An ahp-based flexible relay node selection scheme for wbans. *Wireless Personal Communications*, pages 1–20, 2016.
  - [9] Rongrong Zhang, Hassine Moungla, and Ahmed Mahaoua. A reliable and energy-efficient leader election algorithm for wireless body area networks. In *Communications (ICC), 2015 IEEE International Conference on*, pages 530–535. IEEE, 2015.
  - [10] Ugo Colesanti and Silvia Santini. The collection tree protocol for the castalia wireless sensor networks simulator. *ETH Zurich, Zurich, Switzerland*, 2011.
  - [11] Euhanna Ghadimi, Olaf Landsiedel, Pablo Soldati, Simon Duquennoy, and Mikael Johansson. Opportunistic routing in low duty-cycle wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 10(4):67, 2014.
  - [12] Sriyanjana Adhikary, Sankhyayan Choudhury, and Samiran Chattopadhyay. A new routing protocol for wban to enhance energy consumption and network

lifetime. In *Proceedings of the 17th International Conference on Distributed Computing and Networking*, page 40. ACM, 2016.

- [13] Reva Kachroo and DR Rohit Bajaj. A novel technique for optimized routing in wireless body area network using genetic algorithm. *Journal of Network Communications and Emerging Technologies (JNCET) www. jncet. org*, 2(2), 2015.
- [14] Harsharan Pal Kaur and Kirtika Goyal. Cost based efficient routing for wireless body area networks. 2015.
- [15] Priya Juneja. *Reliable Energy Efficient Adaptive Routing Algorithm for Body Area Networks*. PhD thesis, THAPAR UNIVERSITY PATIALA, 2015.
- [16] Haider Abd Ali Sabti and David Victor Thiel. Self-calibrating body sensor network based on periodic human movements. *Sensors Journal, IEEE*, 15(3):1552–1558, 2015.

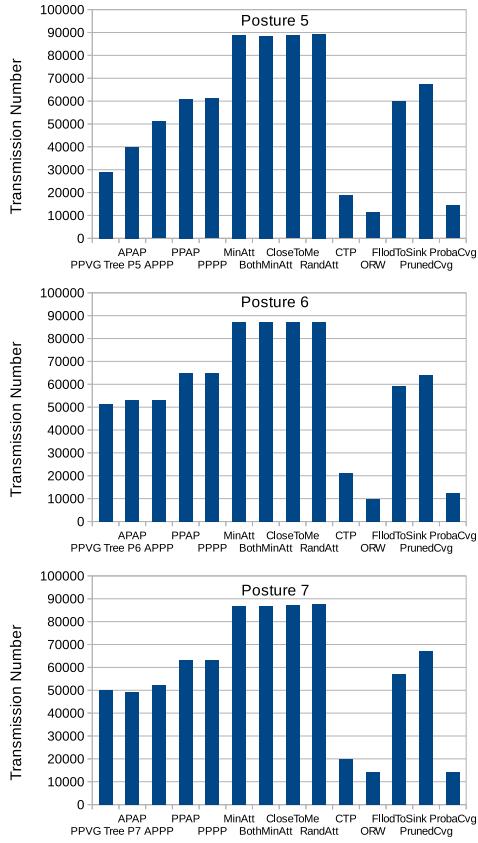


Figure 17: Number of Transmissions comparison among all the strategies in Posture 5-7

- [17] Priya Juneja and Sushma Jain. Tree based energy efficient routing scheme for body area network. In *Computer Engineering and Applications (ICACEA), 2015 International Conference on Advances in*, pages 940–947. IEEE, 2015.
- [18] KS Deepak and AV Babu. Enhancing reliability of ieee 802.15. 6 wireless body area networks in scheduled access mode and error prone channels. *Wireless Personal Communications*, pages 1–26, 2016.
- [19] Madhumita Kathuria and Sapna Gambhir. Reliable delay sensitive loss recovery protocol for critical health data transmission system. In *Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE), 2015 International Conference on*, pages 333–339. IEEE, 2015.
- [20] Heikki Karvonen, Jari Inatti, and Matti Hämäläinen. A cross-layer energy efficiency optimization model for wban using ir-uwb transceivers. *Telecommunication Systems*, 58(2):165–177, 2015.
- [21] Hsueh-Wen Tseng, Ruei-Yu Wu, Yi-Zhang Wu, and Shih-Chun Chou. An efficient cross-layer reliable retransmission scheme for the human body shadowing in ieee 802.15. 6-based wireless body sensor networks. In *Proceedings of the 2015 Conference on research in adaptive and convergent systems*, pages 216–222. ACM, 2015.
- [22] Bin Liu, Yong You, and Chang Wen Chen. Cs-based

reliable transmission strategy for wireless body area networks. In *Signal and Information Processing (ChinaSIP), 2015 IEEE China Summit and International Conference on*, pages 448–452. IEEE, 2015.

- [23] Wei Quan, Jianfeng Guan, Zhongbai Jiang, and Hongke Zhang. I-wban: Enabling information-centric data retrieval in heterogeneous wban. In *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on*, pages 138–143. IEEE, 2015.
- [24] Jun-ichi Naganawa, Karma Wangchuk, Minseok Kim, Takahiro Aoyagi, and Jun-ichi Takada. Simulation-based scenario-specific channel modeling for wban cooperative transmission schemes. *Biomedical and Health Informatics, IEEE Journal of*, 19(2):559–570, 2015.
- [25] Wafa Badreddine, Claude Chaudet, Federico Petrucci, and Maria Potop-Butucaru. Broadcast strategies in wireless body area networks. In *Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 83–90. ACM, 2015.

# Best Effort Broadcast under Cascading Failures in Interdependent Networks

Sisi Duan, Sangkeun Lee, Supriya Chinthavali, Mallikarjun Shankar

Oak Ridge National Laboratory  
Oak Ridge, TN 37831, USA

{duans,lees4,chinthavalis,shankarm}@ornl.gov

## ABSTRACT

We present a novel study of reliable broadcast in interdependent networks, in which the failures in one network may cascade to another network. In particular, we focus on the interdependency between the communication network and the power grid network, where the power grid depends on the communication network for control and the communication network depends on the grid for power. In this paper, we propose a best effort broadcast algorithm to handle crash failures in the communication network that may cause cascading failures, where all the correct nodes deliver the message if the sender is correct. At the core of our work is a fully distributed algorithm for the nodes to analyze cascading failures prior to their presence so that failures can be handled accordingly. Our evaluation results show that the algorithm handles cascading failures with little overhead.

## CCS Concepts

•Networks → Network protocol design; Routing protocols; Network simulations; •Computing methodologies → Simulation evaluation; Self-organization;

## Keywords

Interdependent networks, best effort reliable broadcast, cascading failures, crash failures, soft links

This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (<http://energy.gov/downloads/doe-public-access-plan>).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

ICDCN '17, January 04-07, 2017, Hyderabad, India

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

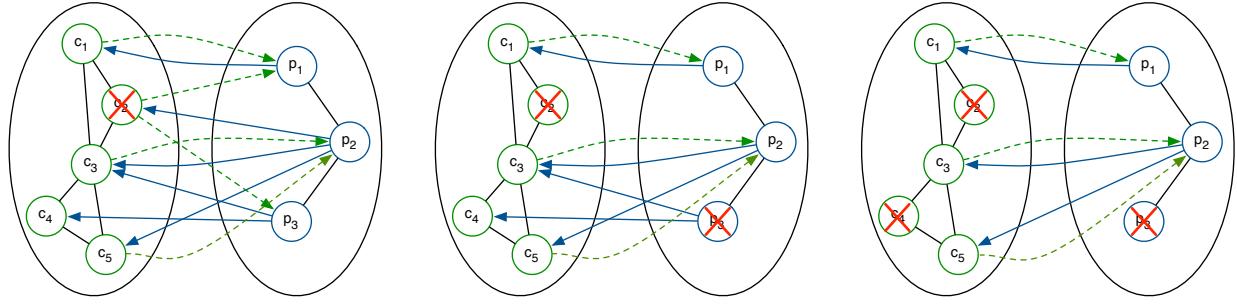
DOI: <http://dx.doi.org/10.1145/3007748.3007752>

## 1. INTRODUCTION

Modern network services are becoming increasingly dependent on infrastructure networks such that a single failure may cascade to another network and cause the failures of all the dependent networks. Such failures may cascade multiple times in a zigzag manner between the networks and cause widespread failures. A particular example is the interdependency between the power grid and the communication network. The 2003 Italian blackout [15], 2003 U.S. Northeastern power outage [1, 6], and 2011 Southwest blackout [2] are all examples of such interdependency. For instance, during the 2003 U.S. Northeastern power outage, 3,175 communication networks suffered from abnormal connectivity outage [6]. A number of previous efforts focus on the analysis of the robustness of interdependent networks. To the best of our knowledge, none of the previous work formalized the problem between the communication network and other interdependent networks and studied resilient solutions to handle such failures.

We study reliable broadcast in a multihop communication network *c*-network and a power grid network *p*-network, which are mutually dependent. The *c*-network is composed of a set of *c*-nodes (e.g., routers, sensors, etc.) connected by communication links and the *p*-network is composed of a set of *p*-nodes (e.g., power substations) connected by power lines. In order for the nodes to operate, a *c*-node must receive power from at least one *p*-node and a *p*-node must receive control signals from at least one *c*-node. We model the interdependency using graphs. As illustrated in Figure 1, when  $c_2$  fails, it cannot provide control signals to  $p_1$  and  $p_3$ . Node  $p_1$  can still operate since  $c_1$  has an edge to it. However,  $p_3$  fails and it cannot provide power to  $c_3$  and  $c_4$ . Node  $c_3$  still operates with power from  $p_2$ . Since no other *p*-nodes have edges to  $c_4$ ,  $c_4$  fails. In the communication network, a *c*-node sends a message through certain paths to some *c*-nodes. *C*-nodes are subject to crash failures, which can be reliably detected by other *c*-nodes. In comparison, correct nodes faithfully follow the protocols. Our goal is to design a solution that guarantees best effort broadcast despite the presence of crashing *c*-nodes, where all the correct nodes deliver the messages if the sender is correct.

We use soft links to handle cascading failures, which are backup links that are activated to handle the failures of primary communication links. The idea of soft links is not new. Specifically, in an independent network, in order to handle the failure of a neighbor, a node  $c_i$  only needs to maintain a soft link to the neighbor of its neighbor so that messages can still be sent along the path. However, in interdependent net-



(a) Node  $c_2$  fails. It cannot provide control signals to  $p_1$  and  $p_3$ . (b) Node  $c_1$  has a link to  $p_1$  and  $p_1$  operates. Node  $p_3$  fails. (c) Node  $p_2$  has a link to  $c_3$  and  $c_3$  operates. Node  $c_4$  fails.

**Figure 1: Cascade of a single failure in the interdependent model between a c-network of 5 c-nodes and a p-network of 3 p-nodes.**

works, since failures occur in a widespread cascading fashion, it is possible that the neighbor of  $c_i$ 's neighbor also fails. In fact, it is extremely challenging to determine how many soft links are enough to handle even one single failure without knowing all the cascading effect. Indeed, without a carefully designed algorithm, nodes cannot analyze the cascading failures to handle them. A straightforward solution is to rely on a powerful and centralized computing agent/node to analyze the failures for each node. However, it will cause large communication and computing overhead for the agent since each node must communicate with the centralized agent to learn the result and maintain soft links. Although it might be possible to build a set of distributed agents, it can still cause high communication and maintenance overhead.

We present a best effort broadcast algorithm where nodes analyze cascading failures and maintain the information in a fully distributed manner. At the core of our approach are two key sub-algorithms: *f-information collection* and *link management*. The *f-information collection* is a communication algorithm for the c-nodes to analyze and collect the information of all the cascading failures caused by a single c-node. Based on the f-information of each neighbor, a c-node can learn the next correct c-node in each path so as to maintain soft links. On the other hand, the *link management* is a mechanism for the nodes to update their routing tables in the presence of failures so that nodes can manage their soft links for long term robustness. As a result, soft links can be correctly maintained and best effort broadcast can be guaranteed if there are no failures during message transmission in the algorithm.

Due to the use of the above approach, best effort broadcast is achieved with the following benefits. First, nodes only need to maintain minimum information in order to analyze the failures. Indeed, since we use a distributed failure analysis algorithm, nodes do not need to maintain the whole network. Second, the information of cascading failures is collected in a fully distributed manner. Last but not least, our algorithm enables very effective usage of soft links. In order to handle one c-node failure, each c-node maintains only one soft link although a set of consecutive c-nodes may fail as a cascading effect. This guarantees that messages can be reliably broadcast to every correct c-node in the communication network. Our evaluation results show that our algorithm achieves low packet drop rate and generates little

overhead to the normal network traffic. The tradeoff is a slightly longer delay in handling failures.

Our paper makes the following contributions:

- We present the first reliable broadcast model in the interdependent networks. We study best effort broadcast in the presence of crash failures in the communication network, which may cause cascading failures in both power grid network and communication network.
- We present a fully distributed algorithm for the nodes to analyze the cascading failures. Each node maintains minimum information for the dependent network.
- We use inactive soft links in addition to the primary links in the communication network to achieve best effort broadcast. In order to handle one failure, each c-node only maintains one soft link although multiple cascading failures may occur due to a single failure.
- Our evaluation results show that our algorithm achieves low packet drop rate and generates little overhead to the normal network traffic. The tradeoff is a slightly longer delay in handling failures.

## 2. RELATED WORK

Modeling interdependences between critical infrastructure networks is challenging due to a wide range of dimensions such as the type of coupling and type of failures [22, 23]. Previous studies of interdependent network systems focus mainly on the analysis of robustness [3, 12–14, 20, 21, 26]. A number of work study the interdependency between communication network and power grid, most of which focus on finding the vulnerabilities of existing network [21] or the design of a robust topology [13]. In comparison, we study a resilient solution that handles the failures in communication network in the interdependency model.

Reliable broadcast [4] has been widely studied that includes several categories such as regular reliable broadcast and uniform broadcast. We study best effort broadcast, where all the correct nodes deliver the message if the sender is correct. In terms of failures, previous studies tolerate both crash failures [18, 24] and Byzantine (arbitrary) failures [4] in both highly connected network [10, 18] and loosely connected network [7, 19]. We study crash failures in the communication network in the model of interdependent networks.

Reliable broadcast of multipath message forwarding has also been studied in publish/subscribe systems [16, 17]. The

use of soft links has been proposed [17] to handle failures during message forwarding. Each node maintains several soft links that can be activated in the presence of failures. We use similar idea of soft links to handle failures.

Failure detectors were proposed previously to detect faulty behaviors [5, 8]. Chandra and Toueg [5] introduced the notion of unreliable failures detectors, where each failure detector outputs the identity of processes suspected to have crashed and nodes can rely on it for message transmission. We also use failure detectors for c-nodes to detect crashing c-nodes in their routing tables.

### 3. INTERDEPENDENCY MODEL

We study the interdependency between two networks: the power grid network *p-network* and the communication network *c-network*. The power grid consists of a set of  $n$  *p-nodes*  $p_1, p_2, \dots, p_n$  (e.g. substations). The communication network consists of a set of  $m$  *c-nodes*  $c_1, c_2, \dots, c_m$  (e.g., routers, sensors, etc.). The p-nodes are connected with power lines and the c-nodes are connected with communication links. Each c-node constantly receives power from the p-nodes and every p-node constantly receives control signals from the c-nodes. We follow a model similar to the one used in previous work [9, 13, 21], where a p-node *operates* if it receives control signals from at least one c-node and a c-node *operates* if it receives power from at least one p-node. We assume c-nodes do not have backup battery, i.e., a c-node immediately fails if it does not receive power from any p-nodes. In addition, we assume that power substations are connected to power generators, i.e., each power substation is connected to a generator that is sufficient for receiving power and we ignore the amount of power supply or demand. In other words, p-nodes can only fail when there are no incoming control signals.

Notation	Meaning
$V_c$	all the c-nodes
$V_p$	all the p-nodes
$E_c$	bidirectional edges between c-nodes
$E_p$	bidirectional edges between p-nodes
$E_{cp}$	directional edges from c-nodes to p-nodes
$E_{pc}$	directional edges from p-nodes to c-nodes
$E_{cp} \& E_{pc}$	interdependency edges
oin degree	number of outgoing interdependency edges
iin degree	number of incoming interdependency edges
$P(\vec{c}_i)$	all the p-neighbors of $c_i$
$P(\vec{c}_i)$	all the p-nodes that have interdependency edges to $c_i$
$C(\vec{p}_i)$	all the c-neighbors of $p_i$
$C(\vec{p}_i)$	all the c-nodes that have interdependency edges to $p_i$
$l_1$	maximum oin degree of any c-node
$l_2$	maximum oin degree of any p-node
$l_3$	maximum iin degree of any p-node
$l_4$	maximum degree of any c-node

Table 1: Notations.

The interdependency between the networks can be represented in a graph  $G = (V, E)$ , as illustrated in Figure 1. We use several notations to represent the network, as shown in Table 1, and we use edges and links interchangeably.  $V =$

$V_c \cup V_p$  is the set of all the nodes and  $E = E_c \cup E_p \cup E_{cp} \cup E_{pc}$  is the set of all the edges. The network is composed of both directional and bidirectional edges to distinguish different features of both individual network and interdependent networks. *Outgoing interdependent network degree* (abbreviated as oin degree) and *incoming interdependent network degree* (abbreviated as iin degree) represent the degree regarding interdependency edges. We also use the term *degree* by default to refer to the number of edges of a node in an individual network. Without loss of generality, we call two c-nodes *neighbors* or *direct neighbors* if there is an edge between them, i.e., they can communicate with each other. The  $E_c$  edges are also called *primary links*. If a c-node  $c_i$  has an edge to a p-node  $p_i$ , we call  $p_i$  a *p-neighbor* of  $c_i$ . Similarly, if a p-node  $p_i$  has an edge to a c-node  $c_i$ , we call  $c_i$  a *c-neighbor* of  $p_i$ .

Correct nodes faithfully follow the communication algorithm and send messages according to the network protocols. Meanwhile, the c-nodes are subject to crash failures but the crashes can be reliably detected by other nodes. We assume fair-loss links, where if a message is sent infinitely often by a correct sender to a correct recipient, it is received infinitely often. Furthermore, links do not produce spurious messages.

We now introduce several notions and define cascading failures.

**DEFINITION 1. (Path)** A sequence of c-nodes  $(c_1, \dots, c_n)$  is a path if,  $\forall i \in \{1, \dots, n-1\}$ ,  $c_i$  and  $c_{i+1}$  are neighbors.

**DEFINITION 2. (Initial Failure)** The failure of a c-node  $c_i$  is an initial failure if its failure is not caused by the loss of incoming interdependency edges.

**DEFINITION 3. (Consecutive Failures)** A sequence of c-nodes  $seq = (c_1, \dots, c_n)$  is a set of consecutive failures if,  $\forall i \in \{1, \dots, n-1\}$ ,  $c_i$  fails,  $n \geq 2$ , and  $seq$  is a path.

**DEFINITION 4. (Single Failure)** The failure of a node  $c_i$  is a single failure if none of its neighbors fails.

**DEFINITION 5. (Cascading Failures)** A number of nodes  $s = (c_1, \dots, c_n, p_1, \dots, p_n)$  are cascading failures if,  $\exists c_i$ , the failure of which makes all the nodes in  $s$  lose incoming interdependency edges.

In other words, we refer to the cascading failures as the failures that are caused by the loss of interdependency edges and the nodes that cause the cascading failures as initial failures. For instance, in the example in Figure 1,  $c_2$  is the initial failure,  $c_2, c_4$ , and  $p_3$  are all single failures, and the set of  $c_3$  and  $p_3$  are cascading failures caused by  $c_2$ . In this paper, we seek to handle single crashing initial failures in the c-network, each of which may cause several cascading failures. For other cases, our algorithm can be further extended to handle failures.

We assume each c-node has a *perfect failure detector*, which provides information about certain c-nodes being crashed or not and it satisfies the following properties.

- *Strong Completeness*. Eventually, every c-node that crashes is permanently detected by every correct c-node.
- *Strong Accuracy*. If a c-node  $c$  is detected by any c-node, then  $c$  has crashed.

The failure detector can be realized using a timeout mechanism. Specifically, in order for a c-node  $c_1$  to detect the correctness of c-node  $c_2$ , it sends a heartbeat message and starts a timer. If  $c_1$  has not received a reply message before the timer expires,  $c_2$  is suspected to be faulty. Although the failure detector abstraction relaxes the timing assumption on nodes and links [4], performance can be guaranteed under partial synchrony [11]: synchrony holds only after some unknown global stabilization timer, but the bounds on communication and processing delays are themselves unknown to the nodes.

We consider the best effort reliable broadcast problem in c-network under the above interdependency model, where all the correct nodes deliver the messages if the sender is correct. It satisfies the following properties.

- *Validity.* If a correct c-node broadcasts a message  $m$  to a set of c-nodes  $DES$ , then every correct c-node in  $DES$  eventually delivers  $m$ .
- *No Creation.* If a c-node delivers a message  $m$  with sender  $s$ , then  $m$  was previously broadcast by  $s$ .

## 4. BEST EFFORT BROADCAST

In this section, we first introduce the preliminaries. Then we introduce our best effort broadcast algorithm. Specifically, in addition to primary links, we also use soft links, which are the information of inactive links to handle the failures of primary links. Through the activation of soft links, new connections are built between correct c-nodes so that messages can be reliably delivered in the presence of failures. Notice that if there exists an alternative path to the destination, re-routing might be an option but it can also consume extra communication overhead and cascading failures may occur in the alternative routes.

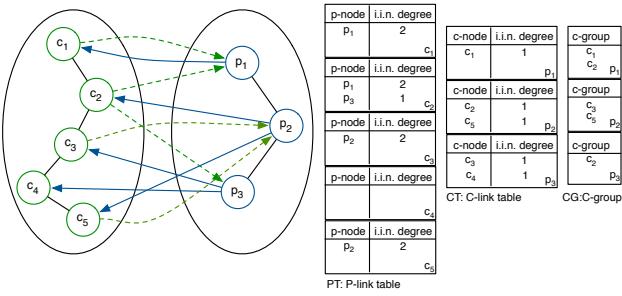


Figure 2: Example of p-link tables, c-link tables, and c-groups.

Our soft link technique is used to guarantee *best effort reliable message transmission* through new communication links so that all the correct destination nodes receive and deliver the same message. In order to correctly build soft links, we employ two sub-algorithms: a cascading failure information collection algorithm called *f-information collection* and an information update algorithm when failures are present, called *link management*. As we will introduce in §4.3, we use a fully distributed message transmission algorithm to analyze cascading failures, based on which nodes can maintain soft links. Lastly, we show link management in §4.4, which is used to update the tables and link information at nodes when failures are detected.

## 4.1 Preliminaries

**Routing Tables.** As illustrated in Figure 2, each c-node maintains several routing tables in the interdependent networks, the definitions of which are shown in DEFINITION 6 to DEFINITION 8. A *p-link table PT* of a c-node consists of all the p-neighbors and the number of their iin degrees. Similarly, for each p-node, there is a *c-link table CT*, which consists of all the c-neighbors and their iin degrees. For each p-node  $p_i$ , all the c-nodes that have interdependency edges to it form a *c-group CG*.

DEFINITION 6. (*P-link table PT*) For any c-node  $c_i$ ,  $PT = \{p_1, \dots\}$  where  $\forall p_j \in PT$ ,  $p_j \in P(\vec{c}_i)$  and  $\forall p_k \in P(\vec{c}_i)$ ,  $p_k \in PT$ .

DEFINITION 7. (*C-link table CT*) For any p-node  $p_i$ ,  $CT = \{c_1, \dots\}$  where  $\forall c_j \in CT$ ,  $c_j \in C(\vec{p}_i)$  and  $\forall c_k \in C(\vec{p}_i)$ ,  $c_k \in CT$ .

DEFINITION 8. (*C-group CG*) For any p-node  $p_i$ ,  $CG = \{c_1, \dots\}$  where  $\forall c_j \in CG$ ,  $c_j \in C(\vec{p}_i)$  and  $\forall c_k \in C(\vec{p}_i)$ ,  $c_k \in CG$ .

We assume all the c-nodes communicate according to routing tables and we refer to it as *regular routing tables*. In other words, for each c-node  $c_i$  and a specific destination  $c_j$ , if  $c_i$  wants to send a message to  $c_j$ ,  $c_i$  looks up its routing table and verify that  $c_j$  is reachable, find a neighbor  $c_k$ , and sends the message to  $c_k$ .

In addition to the regular routing tables for message transmission in a regular communication network, each c-node also maintains a *p-link table PT*, the *c-link tables* for all the p-nodes in  $PT$ , denoted by  $\{CT\}$ , and all the *c-groups* that it is a member of, denoted by  $\{CG\}$ . For instance, as shown in Figure 2, c-node  $c_1$  has a p-link table with  $p_1$  in it, the c-link table for  $p_1$ , and one c-group with  $c_1$  and  $c_2$ . Similarly, c-node  $c_2$  has a p-link table with  $p_1$  and  $p_3$ , the c-link tables for both  $p_1$  and  $p_3$ , and a c-group, which only has  $c_2$  in it. All the tables can be obtained initially through a heuristic method. We ignore the details in this paper since it is not the main focus of our work.

It is straightforward to see that the number of entries in a p-link table is at most  $l_1$  and the number of entries in a c-link table is at most  $l_2$ , according to the notations in Table 1. Also, a c-node has at most  $l_1$  c-groups and the number of c-nodes in each c-group is at most  $l_3$ . Therefore, in addition to the routing information in a single communication network, the extra storage space for each c-node is  $O(l_1 + l_1 l_2 + l_1 l_3)$ , where  $l_1$  is the size its p-link table,  $l_1 l_2$  is the size of c-link tables for the p-nodes, and  $l_1 l_3$  is the size of all the c-groups. In the worst case,  $l_1$  can be as large as  $n$  and  $l_2$  and  $l_3$  can be as large as  $m$ . Therefore, the storage space complexity is limited by  $O(mn)$ .

**Links.** A c-node has two types of links: primary links and soft links. Primary links of a c-node are the communication links to the neighbors in the c-network in order to perform message transmission. When the primary links fail, soft links are activated and connections are built with the corresponding c-nodes and messages are transmitted to the activated soft links. Each soft link handles one initial c-node failure where the c-node may cause multiple cascading failures. For simplicity, in this paper, for each c-node, we build one soft link to handle one initial failure of one neighbor for

each of its path. If there are consecutive initial failures in each path, the algorithm can be further extended to handle the failures.

**Failure Detector.** As mentioned in §3, each c-node has a built-in failure detector module that provides information about whether certain c-nodes have crashed or not. A c-node uses the failure detector to monitor the correctness of its neighbors (c-nodes), the c-nodes in its c-link tables, and all the c-nodes in its c-groups. Notice that the c-nodes in the c-link tables and c-groups may not be direct neighbors of a c-node. Therefore, the correctness of those c-nodes relies on the neighbors of those c-nodes. Specifically, if a node  $c_i$  wants to learn the correctness of  $c_j$ , which is not its direct neighbor, it can learn the correctness of  $c_j$  from the neighbors of  $c_j$ . This is because each node must monitor the correctness of its neighbors. However,  $c_i$  does not need to maintain the information of the neighbors of  $c_j$ . Instead, it simply sends a  $[fd, c_i, c_j]$  message to  $c_j$ . When a neighbor of  $c_j$  receives this message, it returns the result to  $c_i$ . This applies to all the cases when a c-node needs to monitor a node that is not its direct neighbor. Also notice that if  $c_j$  is not reachable by  $c_i$ , it cannot detect the failures.

## 4.2 Best Effort Broadcast Algorithm

Our best effort broadcast algorithm proceeds as follows. We use a fully distributed f-information collection algorithm (as we will introduce in details in §4.3) for each c-node  $c_i$  to collect the information of cascading failures initialized by  $c_i$ . Since the f-information collection algorithm is initialized by each c-node to analyze all the cascading failures caused by itself, the analysis result is then sent to all the neighbors. Based on such information, each c-node  $c_i$  can learn in each path whether the failure of its neighbor will cause other failures in the path. It can then maintain a soft link to the next correct c-node  $c_j$  in each path. The soft link contains the necessary information for  $c_i$  to build a connection with  $c_j$  and the actual connection is made only when the soft link is activated. Notice that in a single communication network, for each soft link of node  $c_i$ , it only maintains the information of the neighbor of its neighbor, i.e., the c-node that is two hops away. However, in interdependent networks, if the c-node  $c_j$  that is two hops away from  $c_i$  will also fail,  $c_i$  also needs to learn the identity of  $c_k$ —the subsequent node of  $c_j$ , and analyzes whether  $c_k$  will also fail. This process continues until  $c_i$  learns the next correct c-node in the path. For instance, if  $c_i$  learns from its neighbor  $c_j$  that if  $c_i$  and  $c_j$  fails, the subsequent node  $c_k$  of  $c_j$  will also fail but the subsequent node  $c_l$  of  $c_k$  will be correct,  $c_i$  can then build a soft link with  $c_l$  in order to handle the failure of  $c_j$ . In this case,  $c_i$  also needs to learn from  $c_k$  the identity of  $c_l$  since  $c_i$  does not have the information beforehand. After soft links are activated, they become the primary links and the primary links are discarded. New soft links will be maintained after another round of f-information collection.

In the normal case when there are no failures, c-nodes use their regular routing tables for message transmission. If a c-node  $c_i$  detects the failure of its subsequent node  $c_j$  through the failure detector and it has a pending message to  $c_j$ , it first diagnoses the situation using the pre-collected f-information from  $c_j$ . If the destination node  $c_l$  will also fail if  $c_j$  is faulty, it simply stops broadcasting the message since  $c_l$  will also be faulty. Otherwise,  $c_i$  activates the corresponding soft link, builds the connection, and sends messages to the

activated soft link.

For long term robustness, it is important for nodes to monitor the correctness of the c-nodes in the routing tables to maintain the most up-to-date topology. As we will describe in details in §4.4, link management is used to update the tables. The soft links will be updated through another round of f-information collection.

Note that in a mesh topology, it is possible that there are alternative paths to the destination, which we can use to guarantee message delivery. However, it is also possible that failures cascade to the alternative paths. Therefore, the use of soft links is very effective in guaranteeing that all the correct nodes deliver the message as a bottom line. Also note that if some nodes are not initially reachable, the use of soft links is not effective to guarantee best effort broadcast.

```

1 initialization:
2 {CT'}, PT' ← {CT}, PT
3 X.add( $c_i$ )   {output: a set of nodes that will fail}
4 watchlist()
5 for  $p_x$  in PT'
6    $p_x.(in\ degree) \leftarrow p_x.(in\ degree) - 1$ 
7   if  $p_x.(in\ degree) = 0$  then
8     for  $c_k$  in {CT'}. $p_x$ 
9        $c_k.(in\ degree) \leftarrow c_k.(in\ degree) - 1$ 
10      if  $c_k.(in\ degree) = 0$  then
11        watchlist.add( $c_k$ )
12 if !watchlist().empty()
13   send [fcollect,  $c_i$ , X, {CT'}, PT'] to watchlist().first
14 on receiving [fcollect,  $c_{ini}$ , X, {CT'}, PT'] from  $c_j$ 
15 {CT''} ← merge({CT'}, {CT})
16 PT'' ← merge(PT', PT)
17 X.add( $c_i$ )
18 con ← False
19 prev( $c_{ini}$ ) =  $c_j$ 
20 for  $p_x$  in PT"
21    $p_x.(in\ degree) \leftarrow p_x.(in\ degree) - 1$ 
22   if  $p_x.(in\ degree) = 0$  then
23     for  $c_k$  in {CT''}. $p_x$ 
24        $c_k.(in\ degree) \leftarrow c_k.(in\ degree) - 1$ 
25       if  $c_k.(in\ degree) = 0$  then
26         watchlist.add( $c_k$ )
27         con ← True
28   if !watchlist().empty() then
29     send [fcollect,  $c_{ini}$ , X, {CT''}, PT'] to watchlist().first
30   if con = False then
31     send [freturn,  $c_{ini}$ , X, {CT''}, PT''] to prev( $c_{ini}$ )
32 on receiving [freturn,  $c_{ini}$ , X', {CT''}, PT''] from  $c_j$ 
33 if  $c_j = watchlist().first$  then
34   watchlist().remove( $c_j$ )
35   X ← merge(X, X')
36   if watchlist().empty() then
37     send [freturn,  $c_i$ , X, {CT''}, PT'] to prev( $c_{ini}$ )
38   else
39     send [fcollect,  $c_i$ , X, {CT''}, PT'] to watchlist().first

```

**Figure 3: F-information collection algorithm, where  $\{CT\}.p_x$  denotes the c-link table for  $p_x$  in set  $\{CT\}$ .**

## 4.3 F-information Collection

F-information collection is for a c-node  $c_{ini}$  to collect the information of the cascading failures by analyzing its fail-

ure. In this case,  $c_{ini}$  is called the initial failure. This f-information collection algorithm is a distributed message transmission algorithm, as shown in Figure 3. There are two types of messages, *fcollect* from  $c_{ini}$  to the nodes that will fail if  $c_{ini}$  fails, and *freturn* back to  $c_{ini}$  when no further cascading failures will occur.

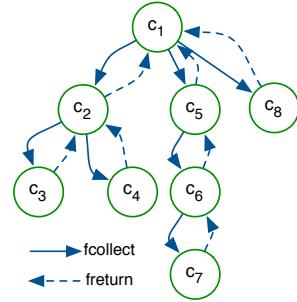
The algorithm proceeds as follows. For each initial node  $c_{ini}$ , the output is a set of cascading failures  $X$ . In the beginning,  $c_i$  (the initial failure) first copies its c-link tables and p-link table, as shown in line (ln) 2, and adds itself to  $X$ . Next, It looks up its p-link table and updates the entries by decreasing the iin degree by 1, as shown in ln 5-6. If any p-node  $p_x$  has an incoming interdependency degree of 0, indicating that if  $c_i$  fails then  $p_x$  will also fail,  $c_i$  starts to lookup the c-link table of  $p_x$ , as shown in ln 7-8. Similarly, it also updates the c-link table by decreasing the iin degree by 1, as shown in ln 9. If any c-node  $c_k$  in the c-link table of  $p_x$  has iin degree of 0,  $c_k$  will also fail if  $c_i$  fails and  $c_k$  is added to the *watchlist()*. A message with  $c_i$  as the *initial node* that initializes the f-information collection,  $X$ ,  $\{CT'\}$ , and  $PT'$  is sent to the nodes in *watchlist()* sequentially, i.e., the message is sent to one node in the *watchlist()* at a time, as shown in ln 12-13. Note that the p-link table and c-link tables are updated on the copies of the original tables and the goal is to mimic the effect of node failures.

When a node  $c_i$  receives such a message with  $X$ ,  $\{CT'\}$ , and  $PT'$ , it first adds itself to the set  $X$  and copies its c-link tables and p-link table. It also merges  $\{CT'\}$  and  $PT'$  to its tables and updates the common entries, as shown in ln 15-16. The purpose of this step is to let nodes analyze the cascading effect, taking into consideration previous failures. For instance, if  $c_1$  and  $c_2$  both have an edge to  $p_1$  and  $c_1$  wants to analyze the cascading failures when it fails, it will not include  $p_1$  in the cascading failures since  $p_1$  still has an incoming edge when  $c_1$  fails. However, if in the following  $c_2$  fails from the failure of another p-node (but also caused by the initial failure of  $c_1$ ), it must include  $p_1$  into the cascading failures since now  $p_2$  has no incoming edges. Therefore, each node must include its copies of c-link tables and p-link tables and each c-node must merge the tables from previous nodes so as to analyze all the failures. The nodes run the same algorithm to analyze the next c-node failures, as shown in ln 20-29. When a c-node will not cause any further cascading failures of c-nodes, it sends a *freturn* message with  $X$  to the previous node, as shown in ln 30-31. This process continues until the message reaches the initial node. When the initial node learns the set of cascading failures when it fails, it sends the f-information to all its neighbors.

In the f-information collection algorithm, each c-node only keeps partial information about the cascading failures, i.e., it receives the *fcollect* from a c-node, computes the subsequent failures, and sends to the corresponding c-nodes that will fail subsequently and waits for *freturn* messages. We use a *watchlist()* scheme for the nodes to collect the information during such a process. This can be represented as a logical Depth First Search (DFS) tree, as shown in Figure 4, where the arrows between nodes represent the message flow and the links between parent and its child nodes may not be real communication links. Instead, the logical tree just demonstrates the sequence of message transmission during f-information collection. The root of the tree is the initial node  $c_{ini}$  that starts the f-information collection process, which is included in both *fcollect* and *freturn* messages.

When a node  $c_i$  receives a *fcollect* message, it keeps the previous node who sent the *fcollect* message (the parent in the tree, which may or may not be the initial node), as shown in ln 19) and watches all the c-nodes that will fail after it fails, i.e., the child nodes in the tree. It sends the *fcollect* to one node in its *watchlist()* at a time and waits for the *freturn* messages. When  $c_i$  receives a *freturn* message, it removes the node from *watchlist()*, as shown in ln 34, and merges  $X'$  to  $X$ , as shown in ln 35. If there are still nodes in its *watchlist()*,  $c_i$  continues to send *fcollect* message until it receives *freturn* from all of them. When there is no node in *watchlist()*, it sends a *freturn* message to its previous node  $prev(c_{ini})$ . This mechanism is necessary for each node to collect all the cascading failures since each node only carries partial information.

**An Example.** As shown in Figure 4,  $c_1$  initializes the f-information collection, where  $X = \{c_1\}$  and  $watchlist() = \{c_2, c_5, c_8\}$ . It first sends a *fcollect* to  $c_2$  and  $c_2$  further sends a *fcollect* message with  $X = \{c_1, c_2\}$  to  $c_3$  and it has  $watchlist() = \{c_3, c_4\}$ . When  $c_2$  receives *freturn* message from  $c_3$  with  $X = \{c_1, c_2, c_3\}$ , it adds  $c_3$  to its  $X$ . Now  $c_2$  has  $watchlist() = \{c_4\}$  and it sends a *fcollect* to  $c_4$ . When  $c_2$  receives  $X = \{c_1, c_2, c_3, c_4\}$  from  $c_4$ , it adds  $c_4$  to its  $X$  and the *watchlist()* becomes empty. It can then send a *freturn* message to  $c_1$  and  $c_1$  has  $watchlist() = \{c_5, c_8\}$ . Similarly for other branches, each message only contains partial information and each node needs to watch its child nodes one by one until it learns results from all of them. Eventually,  $c_1$  learns all the results and the f-information collection is completed.



**Figure 4: Example of f-information collection.**

**The Watchlist.** Notice that we use a sequential mode for nodes to send and collect f-information, i.e., each node sends *fcollect* to one node in its *watchlist()* at a time. This is because each message only contains partial information until the information reaches the last node in the tree ( $c_8$  in the example). Due to the use of the sequential mode, we avoid the case where some failures are not included if the messages are transmitted in parallel to the nodes in the *watchlist()*. For instance,  $c_4$  and  $c_6$  both have an edge to a p-node  $p_9$ . If the *fcollect* messages are sent concurrently to all the child nodes from  $c_1$ , none of  $c_4$  and  $c_6$  will consider the failure of  $p_9$  and therefore some failures may be ignored during this process. In addition, the sequence of sending messages to nodes in the *watchlist()* does not affect the result since eventually all the nodes that will fail will be visited.

Additionally, it is possible that some node in the *watchlist()* is not reachable. In this case, the c-node just skips the node and sends the message to next node in the *watchlist()*. The

reason is that we analyze the failures prior to their presence. For instance, if a node  $c_1$  is not reachable at some node  $c_2$  during the f-information collection process, node  $c_1$  must not be reachable for the initial node  $c_3$  due to the fact that  $c_2$  is reachable for  $c_3$ . Therefore, if some nodes are not reachable prior to the failures, the use of soft links is not effective and it is out of scope of this paper.

#### 4.4 Link Management

In order for c-nodes to maintain the routing tables that represent the most up-to-date topology, each c-node  $c_i$  monitors the correctness of c-nodes in addition to its direct neighbors. These c-nodes include the c-nodes in the c-link tables and c-nodes in all the c-groups of  $c_i$ . The algorithm is shown in Figure 5, where  $\{CG\}.c_j$  denotes all the c-groups that  $c_j$  is a member of,  $cg.p$  is the p-neighbor of the c-nodes in c-group  $cg$ ,  $PT(p)$  is the entry for p-node  $p$  in the p-link table, and  $|cg|$  is the number of nodes in c-group  $cg$ . The goal of monitoring the c-nodes in the c-groups is to update the p-link tables since the incoming interdependency degree of the p-nodes must be updated. The goal of monitoring the c-neighbors of the p-neighbors of  $c_i$  is to update the c-link tables.

```

1 on event  $c_j$  is faulty
2   if  $c_j$  in  $\{CT\}$  then
3     for  $ct$  in  $\{CT\}.c_j$ 
4        $ct.remove(c_j)$ 
5   if  $c_j$  in  $\{CG\}$  then
6     for  $cg$  in  $\{CG\}.c_j$ 
7       if  $c_i = cg.leader$  then
8         send  $[cgupdate, c_j]$  to  $cg$ 
9       else
10        send  $[le, c_j, c_k]$  to  $cg$ 
11         $PT(cg.p).(iin\ degree) \leftarrow PT(cg.p).(iin\ degree) - 1$ 
12        if  $|cg| = 1$  then
13          send  $[sf, c_i, cg.p]$  to  $\{CT\}.(cg.p)$ 
14      if  $c_j, p$  in  $F$  then
15         $CT(p).(iin\ degree) \leftarrow CT(p).(iin\ degree) - 1$ 
16 on receiving  $[cgupdate, c_j]$ 
17    $p_i \leftarrow \{CG\}.c_j.p$ 
18    $PT(p_i).(iin\ degree) \leftarrow PT(p_i).(iin\ degree) - 1$ 
19 on receiving  $[sf, c_j, p]$ 
20    $F.add(c_j, p)$ 
```

**Figure 5: Link management algorithm.**

The key idea for the failure detection is that if a c-node fails, we must remove the interdependency edges and update the tables at all the applicable c-nodes. When the outgoing interdependency edges of a c-node are removed, the corresponding number in the p-link table must be updated, i.e., the iin degree of the p-node in all the applicable p-link tables must be decreased by one. Therefore, we introduce the idea of c-groups and we now introduce the maintenance of c-groups using a leader-based scheme.

In each c-group, a leader is elected and agreed by all the c-nodes. Initially, there is a default leader in each group. The leader monitors the correctness of all the c-nodes in the same group. When it detects the failure of some c-node  $c_j$ , it updates its p-link table and notifies other c-nodes, as shown in ln 7-8. Other c-nodes then simply update their p-link tables, as shown in ln 16-17. If a c-node  $c_i$  is not the leader

in a c-group, it monitors the correctness of the leader. If the leader fails,  $c_i$  notifies all the nodes in the c-group with the id of the new leader  $c_k$ , as shown in ln 9-10. The leader is elected according to the ids in a deterministic rotating manner. When a node receives or has sent a  $[le]$  message, it stores the information of the new leader. If the node is the new leader, it sends a message to all the nodes in the c-group and starts monitoring the correctness of them. In addition, all the nodes also update their p-link tables since the previous leader has failed.

On the other hand, when  $c_i$  fails, we must remove the incoming edges. Therefore, it is straightforward for a c-node  $c_j$  to monitor the c-nodes in its c-link tables since the c-link tables it maintains are the c-link tables of the p-nodes in its p-link table. In this case, if a c-node  $c_i$  in the c-link table(s) fails,  $c_j$  simply removes the corresponding entry, as shown in ln 4 in Figure 5.

If the failure of the  $c_i$  will cause the failures of some p-nodes but the failure will not cascade to the c-network again, we should also update the c-link tables for all the applicable c-nodes. For this purpose, we add another message type called  $[sf]$  where if node  $c_i$  becomes the only node in a c-group  $cg$ , it sends a  $[sf]$  message to the nodes in the c-link table of node  $cg.p$  (the p-neighbor of c-nodes in  $cg$ ), as shown in ln 12-13. When a node  $c_i$  receive a  $[sf]$  message from some node  $c_j$ , it starts monitoring the correctness of  $c_j$  and also adds  $c_j$  to a set  $F$ , as shown in ln 19-20. If it detects the failure of a node in  $F$ , it decreases the degree of  $c_j$  in the c-link table, as shown in ln 14-15.

Assume that a c-node can be the leader of at most  $t$  c-groups, the number of c-nodes a c-node  $c_i$  needs to monitor is limited by  $O(l_4 + l_1l_2 + tl_3)$ , where  $l_4$  is the maximum number of neighbors  $c_i$  needs to monitor,  $l_1l_2$  is the maximum number of c-nodes in the c-link tables of  $c_i$ , and  $tl_3$  is the maximum number of c-nodes  $c_i$  needs to monitor in its c-groups. Since  $l_1$  and  $t$  can be as large as  $n$  and  $l_2$  to  $l_4$  can be as large as  $m$ . The number of c-nodes a node needs to monitor is limited by  $O(mn)$ .

#### 4.5 Correctness

We show the correctness of our best effort broadcast algorithm in the following theorem and we briefly show the proof. In the theorem, *new failures* refer to the failures of nodes involved in the f-information collection and link management process. Our approach guarantees correctness if no failures occur in the f-information. The idea is straightforward. F-information collection is used to analyze the cascading failures. Failures during the algorithm will cause inaccurate results and soft links may not be correctly maintained.

**THEOREM 1.** *Let there be no consecutive initial failures in each path. Best effort reliable broadcast is achieved if there are no new failures among the nodes that participate in each f-information collection and link management process.*

**PROOF.** The no creation property is straightforward. Since we assume nodes can only fail by crashing, each message, if received by some c-node, must be generated by the sender, i.e., no creation property is true.

We now show the validity property in two steps. We first show that if soft links are correctly maintained and there are no consecutive failures, messages are delivered to all the correct receivers. Then we show that if there are no new failures during f-information collection and link management,

soft links can be correctly maintained. Based on these, the validity property can be proved.

We assume that there are no consecutive initial failures. Therefore, one soft link for each node, if correctly maintained, is sufficient to guarantee that messages are reliably delivered to the next correct c-node. For each path, by induction, messages can always be sent along the path to the destination. The destination, if correct, will deliver the message according to the algorithm. Since  $c_s$  is correct, it sends the message to the paths to all the correct destinations. Therefore, the statement is true.

Then we show that if there are no new failures during f-information collection, soft links can be correctly maintained. First, during link management, since we use perfect failure detectors, faulty c-nodes will eventually be detected by correct c-nodes. As discussed in §4.4, we update the p-link tables through the use of c-groups and we update the c-link tables by monitoring of c-nodes in the c-link tables and the use of [sf] messages. If there are no failures during message transmission, all the routing tables will eventually be correctly maintained by all the c-nodes to reflect the most up-to-date topology. Second, since the routing tables are correctly maintained, during f-information collection algorithm, each node is able to analyze the failures. The f-information collection algorithm eventually visits all the c-nodes, if they are connected before the failures, that will fail initialized by each c-node. Therefore, if there are no new failures among all the nodes involved in each f-information collection process, f-information collection enables each c-node to analyze the cascading failures. The correctness of the theorem then follows.  $\square$

## 4.6 Discussion

The correctness of soft links is guaranteed if there are no new failures during f-information collection and link management. In large-scale and highly dependent and dynamic networks, it may not be the case. In this case, nodes may maintain out-of-date c-link and c-group tables and cause wrong analysis results. It is also possible that the f-information algorithm halts where a node waits for a *freturn* message but some nodes during message transmission fail. As we will discuss further in §5, this does not guarantee best effort broadcast where some messages are not delivered to all the correct destinations. We can handle this problem by also using alternative routes for message delivery to increase the delivery rate and adding timers during f-information to ensure that the algorithm will end in the presence of failures.

As discussed previously, the storage complexity for each node is  $O(l_1 + l_1l_2 + l_1l_3)$  and the complexity for the number of nodes each failure detector module is  $O(l_4 + l_1l_2 + tl_3)$ . In the worst case, both are limited by  $O(mn)$ . It is not hard to conclude that in the worst case of a highly connected mesh topology or highly dependent networks, a c-node might eventually need to maintain the information of all the networks and monitor the correctness of all the c-nodes. However, in this case, it is less possible that a failure of a c-node will cause multiple cascading failures.

## 5. EVALUATION

We implement and evaluate our algorithm using OMNeT++ network simulation framework [25]. We construct various sizes of graphs and compare the performance of our distributed algorithm (abbreviated as DA) with the Baseline

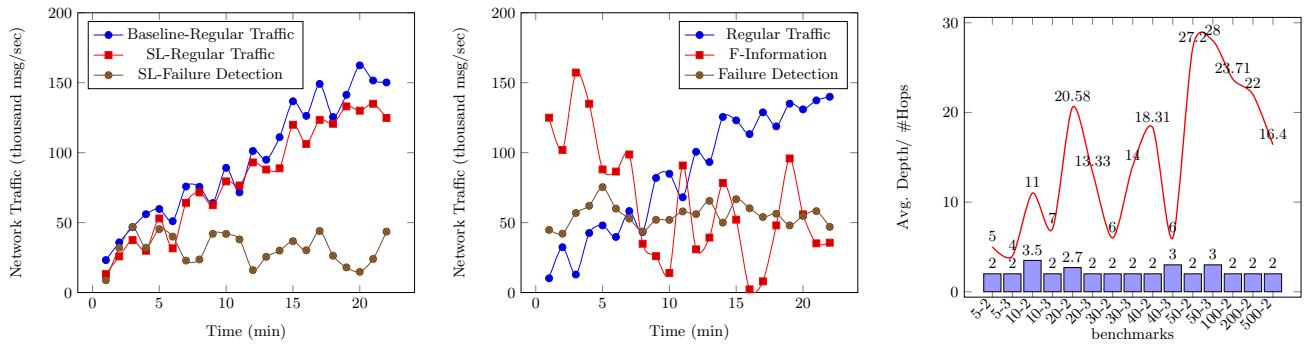
the Soft Link (abbreviated as SF), both in a single communication network. Baseline is a regular routing algorithm where nodes use routing tables for message transmission and do not use soft links. SL builds soft links between nodes that are two hops away. We limit the number of sink nodes to fewer than three and each node generates a packet by doubling the previous period (i.e., 0.01ms, 0.02ms, 0.04ms, etc.). The average delay between two neighbors is set to 0.01ms. When failure detectors are used, each node sends a heartbeat message every 0.3ms and the timer is set to 0.1ms. We set up the maximum outgoing interdependency edges of each node to evaluate the interdependent networks with different dependency level. After the interdependent networks are generated, we check the validity of them by ensuring every node has at least one incoming interdependency edge.

We first assess the network traffic according to message types to evaluate the overhead of our algorithm. We observe that the regular network traffic in SL and our proposed algorithm are in general lower than the Baseline. However, the failure detection and f-information do not decrease the regular traffic to a large degree. We then evaluate the robustness of the algorithm by measuring the percentage of packet drop and the average delay of failure detection. We have proved that best effort broadcast can be guaranteed if there are no new failures during f-information collection and link management. However, as discussed in §4.6, in a large-scale and dynamic network where failures are frequent, new failures can occur and best effort broadcast may not be guaranteed. Therefore, we also evaluate the packet drop rate to assess the efficiency of our algorithm. We notice that our proposed algorithm has largely reduced the packet drop rate and there is a tradeoff of a longer failure detection delay. Lastly, we evaluate the f-information collection delay using various sizes of topologies. We find that due to the way we model the networks, the performance is more related to the number of interdependency edges rather than the degree in c-network. Instead, the average degree of c-network determines how many alternative paths exist for message re-routing. We observe that there is no generic relationship between the number of nodes and the average latency for f-information collection. This indicates that f-information collection process does not increase the overall complexity in a scalable network.

### 5.1 Failure Handling Overhead

We assess the network traffic of different message types and compare our algorithm with Baseline and SL. This is used to assess the overhead caused by our algorithm for distributed failure analysis. In this experiment, we use 200 c-nodes and 200 p-nodes with maximum oin degree of 2. In the c-network, we generate a random mesh graph where the average degree of each node is 3. Each node has 0.1 probability of being crashed. As observed in Figure 6(a), the Baseline algorithm only has regular traffic. Comparably, since SL uses failure detector for each node to monitor the correctness of its neighbors, the regular traffic is in general lower than Baseline. However, the failure detection traffic is relatively stable. This is because each node usually has a fixed number (unless failures occur) of neighbors to monitor.

Compared to Baseline and SL, our proposed DA algorithm generates higher volume of traffic since each node needs to monitor the correctness of a larger number of nodes. Notice that as discussed in §4.6, the c-nodes that a failure detector



(a) Network traffic of different message types for Baseline and SL.  
(b) Network traffic of different message types for the proposed algorithm.  
(c) Average tree depth and average number of hops in f-information collection.

**Figure 6: Evaluation of the algorithms.**

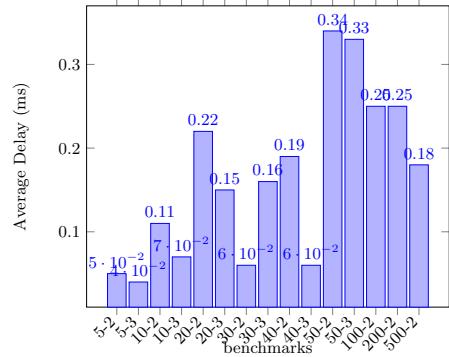
needs to monitor may not be the direct neighbors, where the correctness of the c-nodes is monitored by their neighbors. We count these notification messages also as failure detection traffic. As shown in Figure 6(b), the regular traffic is lower than both Baseline and SL and the failure detection traffic is higher than that of SL. The f-information collection traffic is very high in the beginning. This is because all the nodes need to run f-information collection in the beginning of the experiment. After the initialization, f-information collection is run only when failures occur.

#Nodes	Algorithm	%Packet Drop	Avg FD Delay
50	Baseline	52.19%	N.A.
	SL	28.45%	0.19ms
	DA	3.03%	0.55ms
300	Baseline	51.80%	N.A.
	SL	32.25%	0.21ms
	DA	13.03%	7.14ms

**Table 2: Packet drop rate and average failure detection delay of the algorithms**

## 5.2 Robustness

In order to evaluate the performance of the algorithms under failures, we employ a chain-based topology for c-network with 50 nodes where the nodes are organized sequentially and each node is connected to at most two other nodes. There are 50 p-nodes and the maximum oin degree is set to 2 for both c-nodes and p-nodes. We set up the sink nodes to be the middle of the chain and only nodes in the first half of the chain may fail. As shown in Table 2, since the Baseline does not have a scheme to handle failures, the packet drop rate is high. This can be explained by the fact that each node becomes critical in message transmission. SL has a much lower packet drop rate because it maintains soft links between nodes that are two hops away, which are still effective when the cascading failures do not include too many consecutive failures. Our proposed DA algorithm achieves the lowest packet drop rate since it handles cascading failures. The tradeoff is a slightly longer failure detection delay. Since each node needs to monitor the correctness of a larger number of nodes, the failure detection generates much longer delay due to the communication overhead.



**Figure 7: Average delay for f-information collection.**

We also evaluate the packet drop rate and the average failure detection delay with 300 c-nodes and 300 p-nodes. According to Table 2, the packet drop rate for Baseline is similar with previous case and the packet drop rate for SL slightly increases. This is because in SL each node simply monitors the correctness of its neighbors. Although our proposed DA algorithm still achieves the lowest packet drop rate, the packet drop rate is larger than the case with fewer nodes. Also, since the topology has a larger number of nodes, the failure detection delay is also larger, especially with the nature of chain-based topology where there might be a large number of hops between any two nodes. Therefore, it is not hard to conclude that in highly interdependent and large-scale networks, the failure detection delay and packet drop rate can further be increased.

## 5.3 F-information Collection Delay

In order to evaluate the performance of the algorithm, we assess the average delay of f-information collection process using topologies of various network sizes with 5 to 500 c-nodes and p-nodes. We generate random mesh topologies where the average degree of c-nodes is 3. A benchmark  $x-y$  represents a graph with  $x$  c-nodes,  $y$  p-nodes, and the maximum oin degree is  $y$ . Based on our observation, when the oin degree is bigger than 4, it is less possible that a failure of a c-node causes multiple failures, i.e., soft links between nodes that are two hops away are sufficient. Also, it is straightforward that if each node only has one outgoing

interdependency edge, the networks become highly interdependent, where a single failure causes the failures of almost the whole network. We also show the depth of the tree and the average number of actual hops during each f-information collection in Figure 6(c), which might not be the same with the number of nodes in the tree since the parent node and a child node may not be direct neighbors. We notice that there is not a generic relationship between the number of nodes and the number of hops due to the way we link the nodes. We also notice that the average depth of the tree is 2 to 3. This indicates that an initial failure of a c-node will only cause failures of some c-nodes and the failures will not cascade furthermore. Additionally, as shown in Figure 7, we also evaluate the average latency for f-information collection. Each f-information collection runs for 0.05 to 0.34ms, where the average latency is directly related to the number of actual hops due to the fact that our algorithm essentially visits all the nodes that will fail. This can be explained by the fact that the f-information collection visits the nodes in a DFS manner so that the delay is directly related to the number of nodes that will fail and the distance between them.

## 6. CONCLUSION

In this paper, we study best effort broadcast in the interdependent networks between a multihop communication network and a power grid network. We handle crash failures through the use of soft links in the communication network. In order to efficiently build soft links to handle cascading failures, we present a fully distributed algorithm for the nodes to analyze the failures. Each node needs to maintain minimum extra information, which is updated from time to time to reflect the up-to-date network topology. Based on our evaluation results, our algorithm is effective in handling cascading failures with little overhead.

## 7. REFERENCES

- [1] Final report on the august 14, 2003 blackout in the united states and canada: causes and recommendations. Technical report, U.S. - Canada Power System Outage Task Force, 2004.
- [2] Arizona-southern california outages on september 8, 2011: causes and recommendations. Technical report, Federal Energy Regulatory Commission and the North American Electric Reliability Corporation, 2012.
- [3] S. V. Buldyrev, R. Parshani, G. Paul, H. E. Stanley, and S. Havlin. Catastrophic cascade of failures in interdependent networks. *464:1025–1028*, 2010.
- [4] C. Cachin, R. Guerraoui, and L. Rodrigues. *Introduction to reliable and secure distributed programming*. Springer, 2011.
- [5] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of ACM*, 43(2):225–267, 1996.
- [6] J. H. Cowie, A. T. Ogielsk, B. Premore, E. A. Smith, and T. Underwood. Impact of the 2003 blackouts on internet communications. Technical report, Renesys Corporation, 2003.
- [7] D. Dolev. The byzantine generals strike again. *Journal of Algorithms*, 3(1):14–30, 1982.
- [8] A. Doudou, B. Garbinato, R. Guerraoui, and A. Schiper. Muteness failure detectors: specification and implementation. In *EDCC*, pages 71–87, 1999.
- [9] S. Duan, S. Lee, S. Chinthavali, and M. Shankar. Reliable communication models in interdependent critical infrastructure networks. In *Resilience Week (RWS)*, 2016, pages 152–157. IEEE, 2016.
- [10] S. Duan, H. Meling, S. Peisert, and H. Zhang. BChain: Byzantine replication with high throughput and embedded reconfiguration. In *OPODIS*, pages 91–106, 2014.
- [11] C. Dwork, N. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *Journal of ACM*, 32(2):288–323, 1988.
- [12] J. Gao, S. V. Buldyrev, H. E. Stanley, and S. Havlin. Networks formed from interdependent networks. *Nature Physics*, 8:40–48, 2012.
- [13] M. F. Habib, M. Tornatore, and B. Mukherjee. Cascading-failure-resilient interconnection for interdependent power grid-optical networks. In *OFC*, 2015.
- [14] A. J. Holmgren. Using graph models to analyze the vulnerability of electric power networks. *Risk Analysis*, 26(4):955–969, 2006.
- [15] C. W. Johnson. Analysing the causes of the italian and swiss blackout, 28th september 2003. In *SCS*, 2007.
- [16] R. S. Kazemzadeh and H.-A. Jacobsen. Reliable and highly available distributed publish/subscribe service. In *SRDS*, pages 41–50, 2009.
- [17] R. S. Kazemzadeh and H.-A. Jacobsen. Opportunistic multipath forwarding in content-based publish/subscribe overlays. In *Middleware*, pages 249–270, 2012.
- [18] L. Lamport. The part-time parliament. *ACM Transactions on Computer Systems*, 16(2):133–169, 1998.
- [19] A. Maurer and S. Tixeuil. On byzantine broadcast in loosely connected networks. In *DISC*, pages 253–266, 2012.
- [20] M. Ouyang. Review on modeling and simulation of interdependent critical infrastructure systems. *Reliability Engineering and System Safety*, 121:43–60, 2014.
- [21] M. Parandehgheibi and E. Modiano. Robustness of interdependent networks: the case of communication networks and the power grid. In *GLOBECOM*, pages 2164–2169, 2013.
- [22] P. Pederson, D. Dudenhoefner, S. Hartley, and M. Permann. Critical infrastructure interdependency modeling: a survey of us and international research. *Idaho National Laboratory*, pages 1–20, 2006.
- [23] S. M. Rinaldi, J. P. Peerenboom, and T. K. Kelly. Identifying, understanding, and analyzing critical infrastructure interdependencies. *Control Systems, IEEE*, 21(6):11–25, 2001.
- [24] R. van Renesse and F. B. Schneider. Chain replication for supporting high throughput and availability. In *OSDI*, pages 91–104, 2004.
- [25] A. Varga. OMNeT++. In *Modeling and Tools for Network Simulation*, 2010.
- [26] J. Winkler, L. Dueñas-Osorio, R. Stein, and D. Subramanian. Interface network models for complex urban infrastructure systems. *Journal of Infrastructure Systems*, 17(4):138–150, 2011.

# SER Performance of OFDM-Based AF and DF Cooperative Networks over Asymmetric Fading Channels

Kopal Dwivedi  
 Indian Institute of Technology  
 Patna, Bihar, India  
 kopal.mtc14@iitp.ac.in

Akash Agarwal  
 Indian Institute of Technology  
 Patna, Bihar, India  
 akash.pee15@iitp.ac.in

Preetam Kumar  
 Indian Institute of Technology  
 Patna, Bihar, India  
 pkumar@iitp.ac.in

## ABSTRACT

Cooperative communication provides spatial diversity gain via cooperation of in-cell users, located at different geographical locations that combat fading effects induced by multipath propagation environment. In this paper, the SER performance of OFDM-based Amplify-and-Forward (AF) and Decode-and-Forward (DF) cooperative diversity schemes over asymmetric fading channels has been investigated. The Source-Relay (SR) and Relay-Destination (RD) links are subjected to Rayleigh fading distribution whereas, Source-Destination (SD) link follows generalized Gamma fading distribution. Firstly, SER performance of M-ary Phase Shift Keying (MPSK) has been derived for asymmetric fading channels. Later, the average Symbol Error Rate (SER) of Binary (B)-PSK ( $M=2$ ) and Quadrature(Q)-PSK ( $M=4$ ) are analyzed with different channel parameters. Finally, simulation results are presented to corroborate theoretically derived results.

## CCS Concepts

- Networks → Network performance analysis; Network simulations;

## Keywords

Cooperative Communication, OFDM, Symbol Error Rate (SER), Amplify and Forward (AF), Decode and Forward (DF) relaying, Asymmetric Fading channel.

## 1. INTRODUCTION

Cooperative communication has a leading role for enabling high data rate and coverage over large areas for future generation of wireless communication. Recently, cooperative communication has drawn much research attention because of its potential to increase the capacity of channel, coverage range and also to improve link reliability in wireless network. In cooperative system, the surrounding nodes relay the signal of source terminal to destination, which results

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

ICDCN '17, January 04-07, 2017, Hyderabad, India

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007751>

into spatial diversity. This advantage of spatial diversity is achieved by creating the virtual antenna array through relaying mechanism [1]-[2]. In wireless communication scenario, it is not necessary that all the links would follow the same fading distribution. For instance, one of the available wireless links might be in line of sight (LOS) condition experiencing Rician fading distribution, while the rest of them could be in non-LOS (NLOS) condition following the Rayleigh fading distribution or generalized Gamma distribution, due to random location of relays. Such scenario in cooperative communication is known as asymmetric fading channel.

The idea of cooperative communication was first proposed in 1970s [3]-[4], in which a three node cooperative model was explained in the context of mutual information. Several low-complexity relaying protocols were proposed in [5]. The two most commonly analyzed protocols are AF and DF. In AF protocol, the relay terminal amplifies the received signal from source and then retransmits it to the destination terminal. Whereas in DF protocol, the relay node first decodes the signal, re-encodes it and then forwards it to destined node. The relay channel capacity was first explained and analyzed in [6]. Further, SER performance evaluation of AF and DF protocols over Rayleigh distributed channel has been analyzed by following the moment generating function (MGF) approach presented in [9]. Additionally, performance evaluation of cooperative diversity over the Nakagami-m distributed fading channel was analyzed in [10]. The SER analysis of AF cooperative network over asymmetric channel is presented in [12], considering the combination of Rayleigh-Rician fading channel.

Further, multicarrier schemes with the cooperative communication plays a key role in the performance improvement. The BER performance of OFDM-based AF cooperative relay system is investigated in [13]. The SER performance of two hop transmission system with DF cooperative relaying over Nakagami-m distributed fading channel has been derived in [15]. The approximate average SER analysis of AF-OFDM system considering the non-linear distortions induced by a power amplifier is derived in [16]. The SER expression of the multi-hop and multi-branch AF relay is evaluated in [17]. The repetition-based DF cooperative relay network over asymmetric fading channels is analyzed in [18].

However, to the extent of our knowledge, performance evaluation of OFDM-based AF and DF cooperative relay system over mixed Rayleigh and Nakagami-m fading channel (asymmetric channel) has not been reported yet. Such

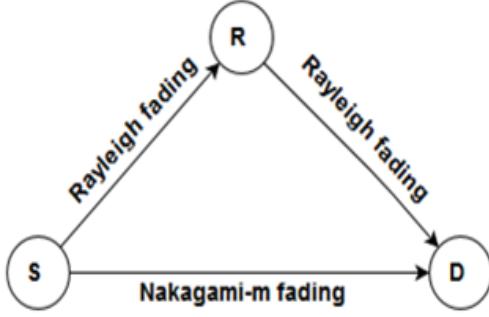


Figure 1: A three-node cooperation signal model

OFDM based asymmetric channel analysis would be helpful in investigating the 5G communication scenario in future. In this paper, the exact average SER expressions of OFDM-based AF and DF cooperative relay protocols are derived for M-PSK.

The remainder of this paper is organized as follows: In Section 2, we introduce system model used in this paper. In section 3 and 4, the SER expressions for AF-OFDM and DF-OFDM are obtained by calculating MGF of the total end to end Signal-to-Noise Ratio (SNR). The performance evaluations and conclusion are presented in Section 5 and 6 respectively.

## 2. SYSTEM MODEL

### 2.1 OFDM based AF and DF relaying

Considering a two-hop OFDM based cooperative system, source  $S$  with single relay  $R$  and destination  $D$  as shown in Fig 1. A block of OFDM transceiver system with  $N$  subcarriers is present at each terminal, where each subcarrier is assumed to have same transmission power. Frequency and timing synchronization among nodes is assumed. The length of OFDM Cyclic Prefix (CP) has been considered longer than the maximum multipath time delay spread of fading channel.

To maintain orthogonality, Time Division Multiple Access (TDMA) is used to access the wireless channel during the two phases of transmission i.e. broadcasting and cooperation phase. In broadcasting phase (phase 1), source broadcasts the message signal to both relay  $R$  and destination node  $D$  respectively. In cooperation phase (phase 2), relay terminal forwards the received signal to the destination.

Depending upon the above considerations, in phase 1 the discrete-time baseband signal received at destination  $Y_{sd}(n)$ , and relay terminal  $Y_{sr}(n)$  at  $n^{th}$  subcarrier is given as

$$Y_{sd}(n) = \sqrt{P_s} h_{sd}(n) x_s + \eta_{sd}(n) \quad 1 \leq n \leq N \quad (1)$$

$$Y_{sr}(n) = \sqrt{P_s} h_{sr}(n) x_s + \eta_{sr}(n) \quad 1 \leq n \leq N \quad (2)$$

where  $P_s$  is transmitted power from source,  $h_{sd}(n)$  and  $h_{sr}(n)$  are channel coefficients at  $n^{th}$  subcarrier for S-D link and S-R link having variances  $\Omega_{sd}$  (nakagami-m distribution) and  $\delta_{sr}^2$  (Rayleigh distribution) respectively.  $x_s$  is transmitted symbol from the source at  $n^{th}$  subcarrier.  $\eta_{sd}(n)$  and  $\eta_{sr}(n)$  are Additive White Gaussian Noise

(AWGN) components with zero mean and variance  $N_0$ . It is assumed that data symbols  $x_s$  are independent and identically distributed (i.i.d).

In phase 2, by employing AF relay strategy, the node  $R$  amplifies the signal received from source terminal  $Y_{sr}(n)$  by a multiplicative factor  $G_n$ , and forwards amplified signal to the destination with transmitted power  $P_r$ . Amplification gain  $G_n$  can be expressed as [7]

$$G_n = \sqrt{\frac{P_r}{P_s |h_{sr}(n)|^2 + N_0}} \quad (3)$$

where  $N_0$  is the noise variance. The amplification factor fully depends upon channel condition between source and relay terminals. At the destination node (BS), the received signal from relay  $R$  at  $n^{th}$  subcarrier is of the form [6]

$$Y_{rd}(n) = G_n h_{rd}(n) Y_{sr}(n) + \eta_{rd}(n) \quad (4)$$

where  $h_{rd}(n)$  is the channel coefficient of  $n^{th}$  subcarrier of R-D link and  $\eta_{rd}(n)$  is the corresponding AWGN noise term. Final received signal at destination is obtained by substituting (2) and (3) in (4)

$$Y_{rd}(n) = \sqrt{\frac{P_s P_r}{P_s |h_{sr}(n)|^2 + N_0}} h_{rd}(n) h_{sr}(n) x_s + \eta'_{rd}(n) \quad (5)$$

$$\text{where } \eta'_{rd}(n) = \sqrt{\frac{P_r}{P_s |h_{sr}(n)|^2 + N_0}} h_{rd}(n) \eta_{sr}(n) + \eta_{rd}(n)$$

For DF cooperation model, selective relaying protocol is assumed. In this case, when  $R$  decodes received data from  $S$  correctly, only then it forwards the decoded data with power  $P_r$  to the destination node i.e.  $\tilde{P}_r = P_r$ , otherwise  $R$  remains idle i.e.  $\tilde{P}_r = 0$ . Hence, the received signal expression at  $D$  in phase 2 can be written as-

$$Y_{rd}(n) = \sqrt{\tilde{P}_r} h_{rd}(n) x_s + \eta_{rd}(n) \quad 1 \leq n \leq N \quad (6)$$

where  $h_{rd}(n)$  is channel coefficient from relay to destination with variance  $\delta_{rd}^2$ . The noise part  $\eta_{rd}(n)$  is modeled with zero mean and variance  $N_0$ .

For both AF and DF cooperation protocols, it is assumed that the channel coefficients  $h_{sr}(n)$ ,  $h_{sd}(n)$  and  $h_{rd}(n)$  are independent to each other and known at the receiver. Maximal Ratio Combiner (MRC) is used to combine the received signal from both phases (broadcasting and cooperation phase) at BS. The total transmitted power is assumed to be  $P_s + P_r = P_t$ . The total end-to-end SNR at destination at  $n^{th}$  subcarrier is written as [19]

$$\gamma_n^{tot} = \gamma_n^{sd} + \gamma_n^{sr} \quad (7)$$

$$\gamma_n^{tot} = \gamma_n^{sd} + \frac{\gamma_n^{sr} \gamma_n^{rd}}{\gamma_n^{sr} + \gamma_n^{rd} + 1} \quad (8)$$

where  $\gamma_n^{sr} = |h_{sr}(n)|^2 P_s / N_0$  and  $\gamma_n^{rd} = |h_{rd}(n)|^2 P_r / N_0$  are instantaneous SNRs between S-R link and R-D link at  $n^{th}$  subcarrier respectively.  $\gamma_n^{sd} = |h_{sd}(n)|^2 P_s / N_0$  is instantaneous SNR between S-D link at  $n^{th}$  subcarrier.

### 2.2 Channel modelling of asymmetric fading channel

There could be several combinations of asymmetric fading channels. In Fig 1, one such possible case of asymmetric channel has been considered where S-R and R-D links are

subjected to Rayleigh fading condition, whereas S-D link follows Nakagami-m fading condition. The instantaneous SNR at  $n^{th}$  subcarrier  $\gamma_n^{sr}$  and  $\gamma_n^{rd}$  is exponentially distributed. The PDF is given as,

$$f_{\gamma_n^{sr}}(\gamma_n^{sr}) = \frac{1}{\bar{\gamma}_n^{sr}} \exp\left(-\frac{\gamma_n^{sr}}{\bar{\gamma}_n^{sr}}\right) \quad (9)$$

$$f_{\gamma_n^{rd}}(\gamma_n^{rd}) = \frac{1}{\bar{\gamma}_n^{rd}} \exp\left(-\frac{\gamma_n^{rd}}{\bar{\gamma}_n^{rd}}\right) \quad (10)$$

where  $\bar{\gamma}_n^{sr} = E[\gamma_n^{sr}]$  and  $\bar{\gamma}_n^{rd} = E[\gamma_n^{rd}]$  are average SNR of S-R and R-D links respectively.

S-D link experiences Nakagami-m fading distribution, where  $\gamma_n^{sd}$  follows generalized gamma distribution with PDF given as [11]

$$f_{\gamma_n^{sd}}(\gamma) = \frac{\mu(\beta/\bar{\gamma}_n^{sd})^{m\mu}}{\Gamma(m)} (\gamma_n^{sd})^{m\mu-1} \exp\left\{\left(-\frac{\beta\gamma_n^{sd}}{\bar{\gamma}_n^{sd}}\right)^{\mu}\right\}; \\ \gamma_n^{sd} \geq 0 \quad (11)$$

where  $\beta = \frac{\Gamma(m+\frac{1}{\mu})}{\Gamma(m)}$ ,  $m$  denotes nakagami-m fading parameter and  $\mu$  represents shape parameter.  $\Gamma(m)$  is gamma function and  $\bar{\gamma}_n^{sd}$  denotes the average SNR of link SD at  $n^{th}$  subcarrier and given as  $\bar{\gamma}_n^{sd} = E(\gamma_n^{sd})$ .

### 3. SER ANALYSIS FOR AF-OFDM COOPERATIVE RELAY

In AF cooperative diversity, the relay amplifies the signal received from source, where noise also gets amplified as shown in (5). Average SER for M-PSK at  $n^{th}$  subcarrier is given as [20]

$$P_{M-PSK} = \frac{1}{\pi} \int_0^{\frac{(M-1)\pi}{M}} M_{\gamma_n^{tot}}\left(-\frac{d}{\sin^2 \theta}\right) d\theta \quad (12)$$

where  $d = \sin^2 \frac{\pi}{M}$  and  $M_{\gamma_n^{tot}}(s)$  is MGF of total end to end SNR. To find  $M_{\gamma_n^{tot}}(s)$  the upper bound of total end-to-end SNR  $\gamma_n^b$  is of the form [10]

$$\gamma_n^{tot} \leq \gamma_n^b = \gamma_n^{sd} + \gamma_n^{srd} \quad (13)$$

where  $\gamma_n^{srd} = \min(\gamma_n^{sr}, \gamma_n^{rd})$ .  $\gamma_n^{srd}$  denotes instantaneous SNR of composite S-R-D link. MGF of the total end-to-end instantaneous SNR at  $n^{th}$  subcarrier  $\gamma_n^{tot}$  can be written as

$$M_{\gamma_n^{tot}}(s) = M_{\gamma_n^{sd}}(s) \times M_{\gamma_n^{srd}}(s) \quad (14)$$

where  $M_{\gamma_n^{sd}}(s)$  and  $M_{\gamma_n^{srd}}(s)$  are MGF of  $\gamma_n^{sd}$  and  $\gamma_n^{srd}$  respectively. MGF of any non-negative random variable  $\gamma$  is written as [20]

$$M(s) = \int_0^\infty f(\gamma) e^{s\gamma} d\gamma \quad (15)$$

By substituting (11) into (15),  $M_{\gamma_n^{sd}}(s)$  is calculated for the case of  $\mu = 1$  as [11]

$$M_{\gamma_n^{sd}}(s) = \frac{(\beta/\bar{\gamma}_n^{sd})^m}{\Gamma(m)} \int_0^\infty (\gamma_n^{sd})^{m-1} \\ \exp\left\{-\gamma_n^{sd} \left(\frac{\beta}{\bar{\gamma}_n^{sd}} - s\right)\right\} d\gamma_n^{sd} \\ M_{\gamma_n^{sd}}(s) = \left(\frac{\beta}{\bar{\gamma}_n^{sd}}\right)^m \left(\frac{\beta}{\bar{\gamma}_n^{sd}} - s\right)^{-m} \quad (16)$$

Now to calculate  $M_{\gamma_n^{srd}}(s)$ , one needs to find the PDF of  $\gamma_n^{srd}$ . First, CDF of  $\gamma_n^{srd}$  is given as-

$$F_{\gamma_n^{srd}}(\gamma) = 1 - Pr(\gamma_n^{sr} > \gamma) Pr(\gamma_n^{rd} > \gamma) \quad (17)$$

$$F_{\gamma_n^{srd}}(\gamma) = 1 - \left[ \exp\left(-\frac{\gamma}{\bar{\gamma}_n^{sr}}\right) \exp\left(-\frac{\gamma}{\bar{\gamma}_n^{rd}}\right) \right] \quad (18)$$

Now, PDF of  $\gamma_n^{srd}$  is calculated by differentiating (18) with reference to  $\gamma$ , which can be written as

$$f_{\gamma_n^{srd}}(\gamma) = \left[ \frac{1}{\bar{\gamma}_n^{sr}} \exp\left(-\frac{\gamma}{\bar{\gamma}_n^{sr}}\right) \exp\left(-\frac{\gamma}{\bar{\gamma}_n^{rd}}\right) \right. \\ \left. + \frac{1}{\bar{\gamma}_n^{rd}} \exp\left(-\frac{\gamma}{\bar{\gamma}_n^{rd}}\right) \exp\left(-\frac{\gamma}{\bar{\gamma}_n^{sr}}\right) \right] \quad (19)$$

After substituting (19) into (15), the MGF for S-R-D link is written as

$$M_{\gamma_n^{srd}}(s) = \left(\frac{1}{\bar{\gamma}_n^{sr}} + \frac{1}{\bar{\gamma}_n^{rd}}\right) \left(\frac{1}{\bar{\gamma}_n^{sr}} + \frac{1}{\bar{\gamma}_n^{rd}} - s\right)^{-1} \quad (20)$$

By substituting (16) and (20) into (14), the MGF of the total end-to-end instantaneous SNR  $M_{\gamma_n^{tot}}(s)$  can be written as

$$M_{\gamma_n^{tot}}(s) = \left(\frac{\beta}{\bar{\gamma}_n^{sd}}\right)^m \left(\frac{\beta}{\bar{\gamma}_n^{sd}} - s\right)^{-m} \times \\ \left(\frac{1}{\bar{\gamma}_n^{sr}} + \frac{1}{\bar{\gamma}_n^{rd}}\right) \left(\frac{1}{\bar{\gamma}_n^{sr}} + \frac{1}{\bar{\gamma}_n^{rd}} - s\right)^{-1} \quad (21)$$

Now, substituting (21) in (12), an exact SER expression of OFDM-based AF in asymmetric fading channel at  $n^{th}$  subcarrier is obtained as follows-

$$P_{M-PSK} = \frac{1}{\pi} \int_0^{\frac{(M-1)\pi}{M}} \left(\frac{\beta}{\bar{\gamma}_n^{sd}}\right)^m \left(\frac{\beta}{\bar{\gamma}_n^{sd}} - s\right)^{-m} \times \\ \left(\frac{1}{\bar{\gamma}_n^{sr}} + \frac{1}{\bar{\gamma}_n^{rd}}\right) \left(\frac{1}{\bar{\gamma}_n^{sr}} + \frac{1}{\bar{\gamma}_n^{rd}} - s\right)^{-1} d\theta \quad (22)$$

### 4. SER ANALYSIS FOR DF-OFDM COOPERATIVE RELAY

For the DF cooperation, combined signal of Maximal Ratio Combiner (MRC) detector at destination at  $n^{th}$  subcarrier may be written as [22]

$$Y_{MRC} = W_1 Y_{sd}(n) + W_2 Y_{rd}(n) \quad (23)$$

where  $W_1 = \sqrt{P_s} h_{sd}^*(n)/N_0$  and  $W_2 = \sqrt{P_r} h_{rd}^*(n)/N_0$  are MRC coefficients. The output SNR at destination is given as [22]

$$\gamma_n^{MRC} = \frac{P_s |h_{sd}(n)|^2 + \tilde{P}_r |h_{rd}(n)|^2}{N_0} \quad (24)$$

The conditional SER with the arbitrary SNR  $\gamma_n$  for M-PSK modulation is described by the integral [20]

$$P_{MPSK} = \frac{1}{\pi} \int_0^{\frac{(M-1)\pi}{M}} \exp\left(-\frac{d\gamma_n}{\sin^2 \theta}\right) d\theta \quad (25)$$

where  $d = \sin^2(\pi/M)$ .

SER expression for M-PSK with DF relaying can be expressed as [21]

$$\begin{aligned} P_{MPSK}(\gamma_n^{MRC}, \gamma_n^{sr}) &= P_{MPSK}(\gamma_n^{MRC})|_{\bar{P}_r=0} P_{MPSK}(\gamma_n^{sr}) \\ &\quad + (P_{MPSK}(\gamma_n^{MRC})|_{\bar{P}_r=P_r} \times \\ &\quad (1 - P_{MPSK}(\gamma_n^{sr}))) \end{aligned} \quad (26)$$

where  $\gamma_n^{sr} = P_s |h_{sr}(n)|^2 / N_0$  is the instantaneous SNR at  $n^{\text{th}}$  subcarrier at relay terminal. It is assumed that either  $\bar{P}_r = P_r$  or  $\bar{P}_r = 0$ . The probability of erroneous decoding at relay is given as  $P_{MPSK}(\gamma_n^{sr})$ , and the probability that relay decodes correctly is  $(1 - P_{MPSK}(\gamma_n^{sr}))$ . According to (25) the conditional SER in (26) is re-written as,

$$\begin{aligned} P_{MPSK} &= \frac{1}{\pi} \int_0^{\frac{(M-1)\pi}{M}} \exp\left(-\frac{dP_s |h_{sd}(n)|^2}{N_0 \sin^2 \theta}\right) d\theta \\ &\quad \times \frac{1}{\pi} \int_0^{\frac{(M-1)\pi}{M}} \exp\left(-\frac{dP_s |h_{sr}(n)|^2}{N_0 \sin^2 \theta}\right) d\theta \\ &\quad + \frac{1}{\pi} \int_0^{\frac{(M-1)\pi}{M}} \exp\left(-\frac{d(P_s |h_{sd}(n)|^2 + P_r |h_{rd}(n)|^2)}{N_0 \sin^2 \theta}\right) \\ &\quad d\theta \times \left[1 - \frac{1}{\pi} \int_0^{\frac{(M-1)\pi}{M}} \exp\left(-\frac{dP_s |h_{sr}(n)|^2}{N_0 \sin^2 \theta}\right) d\theta\right] \end{aligned} \quad (27)$$

where  $h_{sr}(n)$ ,  $h_{rd}(n)$  and  $h_{sd}(n)$  are channel coefficients of  $n^{\text{th}}$  subcarrier. Averaging the conditional SER (27) over mixed Rayleigh and Nakagami-m distributed fading channels, the SER of the DF cooperation diversity is as follows,

$$\begin{aligned} P_{MPSK} &= F\left(\left(1 + \frac{dP_s \Omega_{sd}}{N_0 m \sin^2 \theta}\right)^{-m}\right) \\ &\quad F\left(\left(1 + \frac{dP_s \delta_{sr}^2}{N_0 \sin^2 \theta}\right)^{-1}\right) \\ &\quad + F\left(\left(1 + \frac{dP_s \Omega_{sd}}{N_0 m \sin^2 \theta}\right)^{-m}\right. \\ &\quad \left.\left(1 + \frac{dP_r \delta_{rd}^2}{N_0 \sin^2 \theta}\right)^{-1}\right) \\ &\quad \times \left[1 - F\left(1 + \frac{dP_s \delta_{sr}^2}{N_0 \sin^2 \theta}\right)^{-1}\right] \end{aligned} \quad (28)$$

where  $F(v(\theta)) = (1/\pi) \int_0^{\frac{(M-1)\pi}{M}} v(\theta) d\theta$ ,  $v(\theta)$  is function with variable  $\theta$ .  $\Omega_{sd}$  is channel variance for S-D link.

#### 4.1 Asymptotic Approximation for SER of OFDM based DF

The SER expression obtained in (28) is mathematically quite complex, and is difficult to evaluate the error performance. From (28), for sufficiently high SNR values or equivalently,  $P_s \Omega_{sd} \gg N_0$ ,  $P_s \delta_{sr}^2 \gg N_0$  and  $P_r \delta_{rd}^2 \gg N_0$ , therefore

$$1 + \frac{dP_s \Omega_{sd}}{N_0 m \sin^2 \theta} \approx \frac{dP_s \Omega_{sd}}{N_0 m \sin^2 \theta} \quad (29)$$

$$1 + \frac{dP_s \delta_{sr}^2}{N_0 \sin^2 \theta} \approx \frac{dP_s \delta_{sr}^2}{N_0 \sin^2 \theta} \quad (30)$$

$$1 + \frac{dP_r \delta_{rd}^2}{N_0 \sin^2 \theta} \approx \frac{dP_r \delta_{rd}^2}{N_0 \sin^2 \theta} \quad (31)$$

Further, it is assumed that  $\Omega_{sd} \neq 0$ ,  $\delta_{sr}^2 \neq 0$  and  $\delta_{rd}^2 \neq 0$ . Hence, (28) can approximated as

$$\begin{aligned} P_{MPSK} &\approx F\left(\left(\frac{dP_s \Omega_{sd}}{N_0 m \sin^2 \theta}\right)^{-m}\right) F\left(\left(\frac{dP_s \delta_{sr}^2}{N_0 \sin^2 \theta}\right)^{-1}\right) \\ &\quad + F\left(\left(\frac{dP_s \Omega_{sd}}{N_0 m \sin^2 \theta}\right)^{-m}\left(\frac{dP_r \delta_{rd}^2}{N_0 \sin^2 \theta}\right)^{-1}\right) \\ &\quad \times \left[1 - F\left(\frac{dP_s \delta_{sr}^2}{N_0 \sin^2 \theta}\right)^{-1}\right] \end{aligned} \quad (32)$$

For sufficiently high SNR values of S-R link [23]

$$\begin{aligned} 1 - F\left(\frac{dP_s \delta_{sr}^2}{N_0 \sin^2 \theta}\right)^{-1} &= 1 - \frac{N_0}{\pi dP_s \delta_{sr}^2} \int_0^{\frac{(M-1)\pi}{M}} \sin^2 \theta d\theta \\ &\approx 1 \end{aligned} \quad (33)$$

Therefore, (32) can be written as,

$$\begin{aligned} P_{MPSK} &\approx F\left(\left(\frac{dP_s \Omega_{sd}}{N_0 m \sin^2 \theta}\right)^{-m}\right) F\left(\left(\frac{dP_s \delta_{sr}^2}{N_0 \sin^2 \theta}\right)^{-1}\right) \\ &\quad + F\left(\left(\frac{dP_s \Omega_{sd}}{N_0 m \sin^2 \theta}\right)^{-m}\left(\frac{dP_r \delta_{rd}^2}{N_0 \sin^2 \theta}\right)^{-1}\right) \end{aligned} \quad (34)$$

On further simplification

$$\begin{aligned} P_{MPSK} &\approx \left(\frac{N_0 m}{dP_s \Omega_{sd}}\right)^m \left(\frac{N_0}{dP_s \delta_{sr}^2}\right) \Delta_{sd} \Delta_{sr} \\ &\quad + \left(\frac{N_0 m}{dP_s \Omega_{sd}}\right)^m \left(\frac{N_0}{dP_r \delta_{rd}^2}\right) \Delta_{sd-rd} \end{aligned} \quad (35)$$

where

$$\Delta_{sd} = \frac{1}{\pi} \int_0^{\frac{(M-1)\pi}{M}} \sin^{2m} \theta d\theta$$

$$\Delta_{sr} = \frac{1}{\pi} \int_0^{\frac{(M-1)\pi}{M}} \sin^2 \theta d\theta$$

and

$$\Delta_{sd-rd} = \frac{1}{\pi} \int_0^{\frac{(M-1)\pi}{M}} \sin^{2(m+1)} \theta d\theta$$

$m$  is an integer value. SER expression derived in (35) is shown to be asymptotically tight in high SNR region in next Section.

## 5. SIMULATION RESULTS

Computer simulations are provided to illustrate the theoretical and simulation results for considered OFDM based cooperative networks. Here the performance of AF and DF cooperation protocols has been analyzed by varying the channel conditions. The analysis for M-PSK modulation with  $M = 2$  (BPSK) and  $M = 4$  (QPSK) for different value of Nakagami parameter  $m$  has been verified through Monte-Carlo MATLAB simulations. The simulation parameters presented in Table 1 are considered for simulations of both AF-OFDM and DF-OFDM cooperative protocols over asymmetric channel.

#### 5.1 Performance of AF-OFDM Cooperation over Asymmetric Channel

Figure 2 and 3 illustrate the simulation and theoretical results for AF-OFDM cooperation with QPSK and BPSK

modulation schemes for different values of Nakagami parameter  $m$ . The transmitted power at source and relay is assumed to be equal, with total transmitted power of unity. Also,  $\delta_{sr}^2 = 1$ ,  $\delta_{rd}^2 = 1$  and  $\Omega_{sd} = 1$  is assumed. It is observed from Fig. 2 and 3 that, the SER approximation from (22) matches well with the simulated results. From figures, it is also observed that by increasing the value of Nakagami parameter  $m$  SER performance improves. The reason behind is that with increasing  $m$ , the power of LOS component of S-D link is higher, which contributes to improved error performance. Here, It is also observed that the performance of QPSK is poorer than BPSK due to larger signal constellation size. At SER of  $10^{-4}$ , a 4 dB gain is achieved with BPSK as compared to QPSK for  $m = 1$  and  $m = 2$ .

## 5.2 Performance of DF-OFDM Cooperation over Asymmetric channel

Figure 4 and 5 shows the SER versus SNR performance of OFDM based DF cooperation with BPSK and QPSK signalling over asymmetric fading environment for different values of  $m$ .  $\delta_{sr}^2 = 1$ ,  $\delta_{rd}^2 = 1$ , and  $\Omega_{sd} = 1$  is considered for simulation. The SER approximation from (35) is asymptotically tight to simulated results in high SNR zone. Again from Fig. 4 and 5, it is observed that by increasing Nakagami factor  $m$  the performance of the considered system is improved in terms of SER, especially for high SNR values. From figures, it could be seen that BPSK modulation outperforms QPSK modulation with the SNR improvement of about 4 dB.

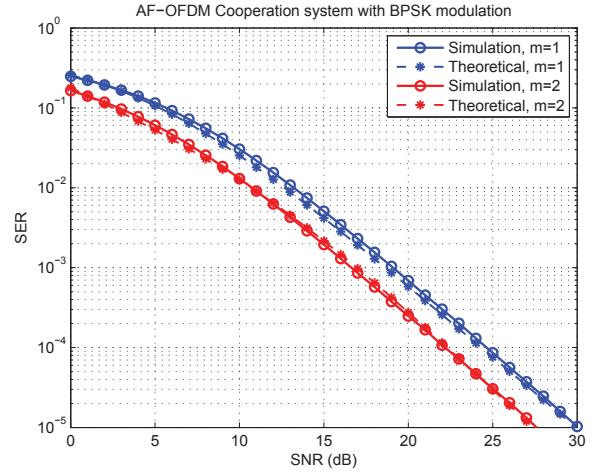
Finally from the simulation results, it is observed that the DF cooperation protocol provides improved SER performance as compared to AF cooperation protocol of about 2 dB at SER of  $10^{-4}$ . Hence, the DF cooperation protocol with BPSK modulation for  $m = 2$  shows the best performance.

**Table 1: Simulation Parameters**

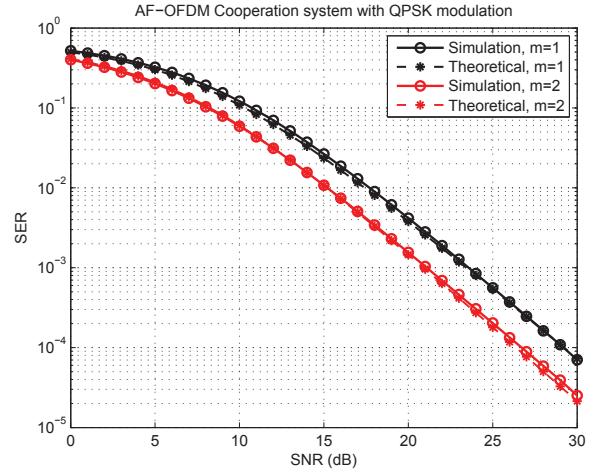
Parameters	Specifications
No. of Subcarriers ( $N$ )	128
Modulation schemes	BPSK and QPSK
Cyclic Prefix	$\frac{N}{4}$ symbol duration
Delay Spread	Uniform (10 taps)
Combining Technique	Maximal-Ratio Combiner

## 6. CONCLUSION

In this paper, we have analyzed the SER performance of two-hop OFDM based AF and DF cooperative diversity with single relay over mixed Nakagami- $m$  and Rayleigh fading channel. The MGF based approach is utilized to analyze the average SER performance, and MRC combining technique is employed at destination to achieve diversity. In the simulation model, the OFDM system with 128 subcarriers, BPSK and QPSK signalling is considered. The power is equally distributed between source and relay terminal. It is illustrated that the analytical and simulation results are in close agreement to each other. From results, it is observed that by using OFDM scheme in cooperative asymmetric relay network the performance is improved. It is also evident that the average SER performance of cooperative DF-OFDM system is better than cooperative AF-OFDM system.



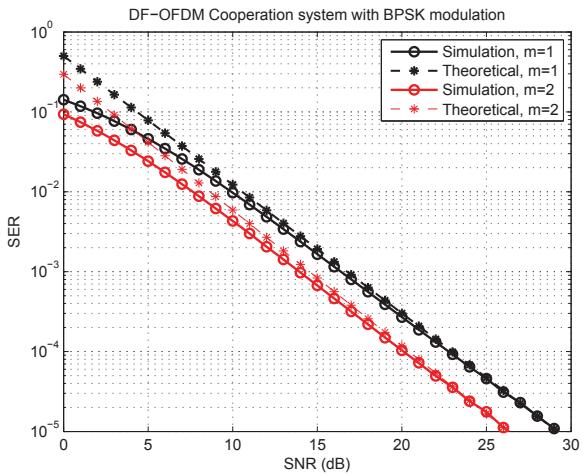
**Figure 2: SER performance of AF-OFDM cooperation with BPSK for  $m = 1$  and  $m = 2$**



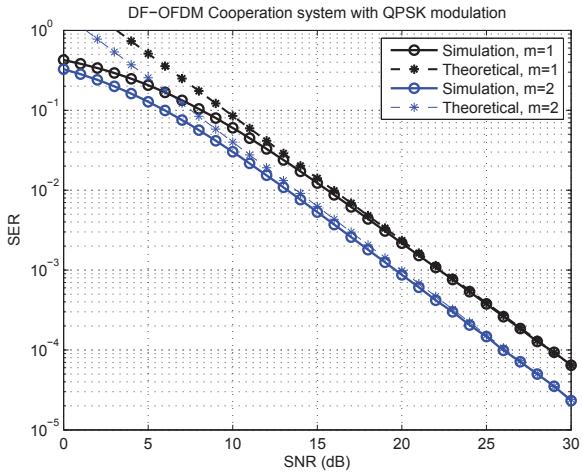
**Figure 3: SER performance of AF-OFDM cooperation with QPSK for  $m = 1$  and  $m = 2$**

## 7. REFERENCES

- [1] A. Sendonaris, E. Erkip, and B. Aazhang. User cooperation diversity. Part I & Part II. System description and Implementation aspects. *IEEE Transactions on Communications*, 51(11):1927-1948, 2003.
- [2] A. Nosratinia, T. E. Hunter, and A. Hedayat. Cooperative communication in wireless networks. In *IEEE Communications Magazine*, 42(10):74-80, 2004.
- [3] Vander Meulen, Edward C. Three-terminal communication channels. *Advances in applied Probability IEEE.*, 3:120-154, 1971.
- [4] Vander Meulen, E. C. A survey of multi-way channels in information theory :1961-1976. *IEEE Transactions on Information Theory*, 23(1):1-37, 1977.
- [5] J. N. Laneman, D. N. Tse, and G. W. Wornell. Cooperative diversity in wireless networks: Efficient protocols and outage behavior. *Information Theory, IEEE Transactions.*, 50(12):3062-3080, 2004.



**Figure 4: SER performance of DF-OFDM cooperation with BPSK for  $m = 1$  and  $m = 2$**



**Figure 5: SER performance of DF-OFDM cooperation with QPSK for  $m = 1$  and  $m = 2$**

- [6] Cover T., Gamal A. E. Capacity theorems for the relay channel. *Information Theory, IEEE Transactions*, 25(5):572-584, 1979.
- [7] A. Kwasinski, Liu KJR. Source channel cooperation tradeoffs for adaptive coded communications. *IEEE Transactions on Wireless Communications*, 7(9):3347-3358, 2008.
- [8] Weifeng Su, A. K. Sadek, K. J. Ray Liu. Cooperative Communication Protocols in Wireless Networks: Performance Analysis and Optimum Power Allocation. *Wireless Pers Commun Springer*, 44:181-217, 2008.
- [9] Anghel, Paul A., and Mostafa Kaveh. Exact symbol error probability of a cooperative network in a Rayleigh-fading environment. *IEEE Transactions on Wireless Commun*, 3(5):1416-1421, 2008.
- [10] S. Ikki and Mohamed H. Ahmed. Performance Analysis of Cooperative Diversity Wireless Networks over Nakagami-m Fading Channel. *IEEE Communication Letter*, 11(4):334-336, April 2007.

- [11] N. Kapucu, M. Bilim, and I. Develi. Ser performance of amplify-and-forward cooperative diversity over asymmetric fading channels. *Wireless personal communications, Springer*, 73(3):1117-1127, 2013.
- [12] J. Ouyang, M. Lin and Y. Zhuang. Performance analysis of cooperative relay networks over asymmetric fading channels. *IET Journals and Magazines Electronics letter*, 48(21):1370-1371, 2012.
- [13] H. A. Suraweera, Jean Armstrong. Performance of OFDM-Based Dual-Hop Amplify-and-Forward Relaying. *IEEE Communication Letter*, 11(9):726-728, September 2007.
- [14] David E. Simmons, Justin P. Coon. Two-way OFDM-based Nonlinear Amplify-and-Forward Relay Systems. *IEEE Transactions on Vehicular Technology*, 2015.
- [15] D. B. da Costa and S. Aissa. Dual-hop decode-and-forward relaying systems with relay selection and maximal-ratio schemes. *Electronics Letters*, 45(9):460-461, 2009.
- [16] C. Alexandre, R. Fernandes, Daniel B. da Costa, Andre L. F. de Almeida. Performance analysis of cooperative amplify-and forward orthogonal frequency division multiplexing systems with power amplifier non-linearity. *IET Commun*, 8(18):3223-3233, 2014.
- [17] M. Mohammadi, Z. Mobini, M. Ardebilipour, B. Mahboobi. Performance Analysis of Generic Amplify-and-Forward Cooperative Networks over Asymmetric Fading Channels. *Wireless Pers. Commun. Springer* 72:49-70, 2013.
- [18] S. Majhi, Y. Nasser, J. Franc. Performance Analysis of Repetition-Based Decode-and-Forward Relaying over Asymmetric Fading Channels. *IEEE, 21st International Symposium on PIMRC*, 362-367, 2010.
- [19] M. O. Hasna, M. S. Alouini. Harmonic mean and end-to-end performance of transmission systems with relays. *IEEE Transactions on Communications*, 52(1):130-135, 2004.
- [20] M. K. simon, M. S. Alouini. Digital communication over fading channels (2nd ed.). *Wiley*, 2000.
- [21] W. Su, A. K. Sadek, and R. J. K. Liu. SER performance analysis and optimum power allocation for decode-and-forward cooperation protocol in wireless networks. *Proc. IEEE WCNC*, 2:984-989, March 2005.
- [22] Brennan, D. G. Linear diversity combining techniques. *Proceedings of the IEEE*, 19(2):331-356, Feb. 2003.
- [23] Yinman Lee, M. H. Tsai. Performance of Decode-and-Forward Cooperative Communications Over Nakagami-m Fading Channels. *IEEE Transaction on vehicular technology*, 58(3):1218-1228, march 2009.

# An NDNoT based Efficient Object Searching Scheme for Smart Home using RFIDs

Divya Saxena

Department of CSE  
IIT Roorkee, India

[divya.saxena.2015@ieee.org](mailto:divya.saxena.2015@ieee.org)

Vaskar Raychoudhury

Department of CSE  
IIT Roorkee, India  
[vaskar@ieee.org](mailto:vaskar@ieee.org)

Christian Becker

Chair for Information Systems II  
Universität Mannheim, Germany  
[Ls-becker@uni-mannheim.de](mailto:Ls-becker@uni-mannheim.de)

## ABSTRACT

Internet of Things (IoT) is driven by innumerable sensing devices, RFID tags and other miniature computing entities, which continuously sense, generate and compute data. One can see that our future environment will contain lots of devices that will generate the massive amount of automated data. The presence of resource constrained devices, frequent exchange of data and heterogeneous types of network traffic distinguish IoT from the current Internet. Recently, Named Data Networking (NDN) is proposed as a content retrieval solution which directly deals with application generated variable-length, location-independent names to search and pull contents for a requesting user, irrespective of hosting entity. The use of content names for communication support name-based routing, in-network caching, and security which make the NDN more suitable for IoT. In this paper, we propose a NDN-based searching mechanism (*named*, Search-NDNoT), which can be used to find any smart item augmented with the RFID in real-world in real-time. An energy-efficient data aggregation algorithm is also proposed to maintain up-to-date data on the server. Our proof-of-concept prototype shows the usability of our proposed system.

## Categories and Subject Descriptors

C.2.1 [COMPUTER-COMMUNICATION NETWORKS]: Network Architecture and Design

## Keywords

Named Data Networking, NDN, IoT, Named Data Networking of Things, NDNoT, ICN, RFID

## 1. INTRODUCTION

Internet-of-Things (IoT) is an environment which connects several devices, sensors, and clouds over the network and allows them to generate and exchange data. IoT allows the use of wireless technologies with the Internet to provide smart functions in the main applications such as, smart home, smart industry, smart health, food, and water monitoring and participatory sensing. Therefore, we can say that our future life will be full of smart objects which can sense, communicate and take the decision by themselves. The Internet of Things allows everyday objects to be identified, tracked, and monitored.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ICDCN '17, January 04-07, 2017, Hyderabad, India  
© 2017 ACM. ISBN 978-1-4503-4839-3/17/01 \$15.00  
DOI: <http://dx.doi.org/10.1145/3007748.3007760>

Current Internet has shown great resilience over the years in connecting the world. But still, IoT suffers from two fundamental challenges [1]. First is, how to provide efficient local and global communication among various heterogeneous IoT-enabled devices. Second is, how to retrieve data from the sensors/actuators in a secure and consistent manner. For the IoT-based systems, robustness, privacy, security, and reliability always remain foremost concerns because of the sensitivity of the data collected, managed and exchanged. There exist several IoT-based systems [2][3][4][5] for identifying and tracking the smart items in the smart home. To find any item, there is a need to identify and then locate them for each user query. For more accuracy, items can be accessed through their names and the whole search area is divided into small localities. All these systems are constrained by the usual host-to-host communication challenges faced by the IoT.

To handle the aforementioned challenges and to implement these systems, Named Data Networking (NDN) can be applied to improve and simplify such IoT communication. NDN is a future Internet paradigm evolved from Information-Centric Networking (ICN) and allowed users to fetch and distribute contents directly using their application-assigned names [6][7]. NDN is free from major host-to-host communication paradigm limitations and have built-in support for security, privacy, reliability and consumer mobility. Using named data relieves NDN from address exhaustion, address assignment, and management problems. NDN can be used as it uses names for applications which remain consistent across facilities and installation. And, the name used once in the architecture remains same for both applications and network infrastructure. NDN uses content signature to secure the content, i.e., identifies every entity in the system by a distinct public key. It uses trust model to authorize an application with a capability and authenticate each and every command for performing required operations on the application. NDN provides a scalable, secure data acquisition platform which collects data from different real-world sensors deployed in real-time. So, we chose to replace *host-to-host communication* with *named data* to leverage the basic characteristics of the later in order to build a robust and scalable object searching algorithm for the smart home.

In this paper, we propose and develop an object searching system using Named Data Networking of Things (NDNoT), named *Search-NDNoT*, which can be used to find any item augmented with RFID in real-world in real-time. NDNoT is a new paradigm where NDN is used as a communication technique.

To the best of our knowledge, *Search-NDNoT* is the first NDN-based object searching mechanism for the smart home. The main contributions of this paper are as follows.

- We propose a novel NDN-based *Search-NDNoT* system for searching the objects for the smart home inhabitants.

- We augment smart items with RFID to identify and locate their location in the smart home environment. To reduce the network traffic, we use the push and pull based communication paradigm for finding item's location. We also propose the data aggregation algorithm to collect data from the network efficiently. As a proof-of-concept, we develop a complete prototype of the system.
- Our experimental result obtained through prototype testbed validates the performance of NDN and usability of our proposed system.

The rest of this paper is organized as follows. In Section 2, we discuss the IoT challenges and solutions provided by the NDN. In Section 3, we discuss the related research works of existing NDNoT-based systems. In Section 4, we describe our proposed NDNoT-based searching scheme for the smart home inhabitants and its algorithm. Section 5 discusses the implementation and testing of the proposed system. We also discuss our performance results supported by thorough analyses. Finally, we conclude the paper in Section 6 and provide directions for possible future extensions.

## 2. BACKGROUND AND MOTIVATION

We first introduce NDN and discuss its applicability to smart environment. We shall also discuss why and how NDN is more suitable for *the things*.

### 2.1 NDN Overview

NDN supports the pull-based delivery model where the content requester ( $U$ ) initiates the communication using a content name ( $CN$ ) [6][7]. NDN uses two types of packets: *Interest packet* ( $I_{pkt}$ ) and *Data packet* ( $D_{pkt}$ ).  $U$  issues an  $I_{pkt}$  for the desired content  $CN$ . When  $I_{pkt}$  reaches to a node having  $CN$ , it issues a  $D_{pkt}$  carrying  $CN$ .  $D_{pkt}$  traces the path of  $I_{pkt}$  in reverse to reach  $U$ . The primary objective of NDN is to deliver valid content to requesters without requiring them to know the location of the hosting entity.

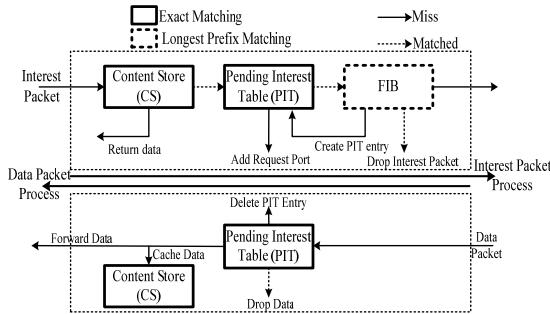


Figure 1. Forwarding process at NDN node [7]

### 2.2 Forwarding in NDN

To forward an  $I_{pkt}$ , each NDN router maintains three data structures, *Content Store* ( $CS$ ), *Pending Interest Table* ( $PIT$ ) [8] and *Forwarding Information Base* ( $FIB$ ) [9][9][10] as shown in Figure 1.  $CS$  is used as a temporary cache for  $D_{pkt}$ (s) to fulfill future requests for already fetched contents. For stateful forwarding, NDN uses  $PIT$  which stores  $I_{pkts}$  detail until they are satisfied.  $FIB$  maintains information about the next-hops and forwards  $I_{pkt}$  using Longest Prefix Match (LPM).

### 2.3 Motivation

The existing Internet/IP-based IoT systems face increasing communication specific limitations. *First*, the plethora of smart

devices are being connected to the IoT will soon deplete the available IP addresses. Limited IPv4 addresses and use of Network Address Translation (NAT) complicates the access and interoperability of these networks. Even though IPv6 can solve the problem to a certain extent, it does not provide any additional security and the application development process is still cumbersome [7]. *Second*, usage of IP addresses renders the system non-adaptive to the addition and/or removal of devices [11]. *Third* is, the Internet requires an additional system to translate application names into IP addresses. *Fourth* is, the Internet relies on the physical or VLAN-based segregation for providing a secure environment for smart sensors, which needs experts to setup and maintain the system. Also, it limits the interoperability amongst the IT systems and renders configurations weak to change [7]. *Fifth*, the Internet frequently fails during data acquisition, and transmission stages because of intermittent connectivity and unreliability of wireless networking. For safety-critical systems, such as healthcare and smart home, this might lead to fatal consequences. *Sixth*, mobility and security are not inherently supported by the Internet and thus, while induced mobility support incurs high hand-off overhead, add-on security features might render privacy-sensitive healthcare system vulnerable.

In order to address the aforementioned challenges, Named Data Networking (NDN) is proposed as a potential alternative networking solution [12]. The main benefits of using NDN in place of Internet are as follows. *First*, NDN uses *CNs* which remain consistent across facilities and installations (over network setup and maintenance), e.g., [/ndn/iitr.ac.in/apps/healthcare/CS/IS\\_lab](http://ndn.iitr.ac.in/apps/healthcare/CS/IS_lab). Naming data provide inherent support for delay-tolerant networking (handles intermittent connectivity) unlike the Internet [6][7]. Moreover, in IoT, as heterogeneous devices are being used, NDN naming can support access restrictions instead of having a separate policy language for individual devices [7]. *Second*, NDN does not face address exhaustion, address assignment, and management problems because it uses unbounded namespaces. *Third*, NDN reduces the complexity of auto-configuration mechanisms compared to layered network architecture [13]. *Fourth*, NDN supports in-network caching, which may reduce power consumption and usage of radio resources depending on the adopted caching strategy. It further supports load balancing and fault tolerance and thus helps resource-constrained devices. *Fifth*, NDN supports multi-path forwarding which provides much sought-after communication reliability for safety-critical systems. *Sixth*, NDN has built-in support for user-side (requester) mobility which will help mobile nodes to keep communicating with the server. *Seventh*, NDN inherently provides confidentiality, authenticity, and integrity [14] by signing each content which is of utmost importance for securing personal data. *Eighth*, NDN uses a trust model to authorize an application and authenticate each and every command (for fixtures) for performing required operations in the application which enhance the privacy, security, and reliability of the system.

## 3. LITERATURE SURVEY

NDN is a promising Internet paradigm, which inherently supports IoT. Zhang, et al. [14] have argued that when communication break among devices due to intermittent connectivity, sleep mode, interference and power off, devices should be able to get the data back after reconnection. Moreover, data generated by a device should be available if the generating device is not connected to the network. The authors show that NDN provides these features

using name-based routing and distributed data caching among a group of devices. Ravindran, et al. [15] have shown the efficient home network based on content-centric approaches. Moreover, they have also compared the features of IP-based and ICN-based home network. Some other research work regarding controlling devices, securing communication and authenticating the user [16][17], have presented IoT for the particular application domains. Zhang, et al. [18] have discussed the main NDN features supporting the IoT more efficiently than the current Internet. Baccelli, et al. [13] have presented the advantages and challenges of the ICN-based approach over IoT. Amadeo, et al. [19] have raised the issues associated with the NDN for supporting the IoT and proposed a new solution for it. Francois, et al. [20] have presented a traffic optimization for IoT in which there is no need to maintain PIT entries for small size data generated by the sensors. Moreover, Saxena, et al. [21] have developed smart healthcare system using NDN overlay over IP.

## 4. SYSTEM ARCHITECTURE

We consider a smart home Named Data Networking of Things (NDNoT) environment where every physical object (e.g., cupboard, microwave oven, key ring, wallet, etc.) has been embedded with RFID tags or sensor nodes to act as an intelligent entity which can be uniquely identified, can perform certain computations and can communicate with its peers wirelessly.

Figure 2 shows the proposed *Search-NDNoT* system architecture. The system works in a hierarchical fashion where the lowest layer entities are the tags, items which move most frequently (e.g. keys, wallet, and cell phone). In most of the cases, a user will be interested in finding one of these items. The second layer consists the entities which are occasionally moved and less frequently (e.g., table) while third layer entities are immovable (e.g. room/office). The fourth layer consists the central server (CSr), a sink of the network which maintains the database of all items with their updated location. We are using hashing to store the locations of items as it provides various benefits over arrays, like faster time access and efficient memory utilization. To identify and track an item, each item is embedded with the sensing device and provided an unique id. Users through their Smartphone can post a query to find any item and see results at the central server (CSr). We have divided each item into two different categories namely, items search using the push-based communication (named, *push-based approach*) and items search using the pull-based communication, (named, *pull-based approach*). CSr stores details and the current location of push tags (fetched using the push-based communication).

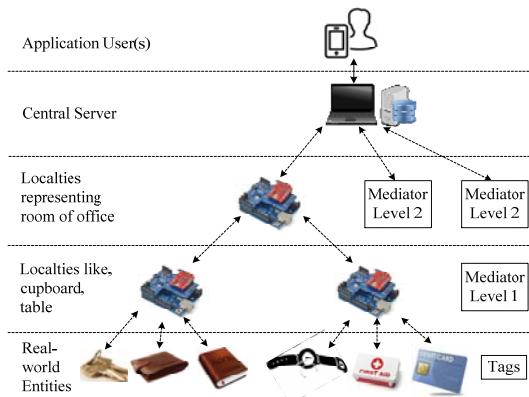


Figure 2. System Architecture

The third layer's devices cover area approximate of a room or office, named as the mediator at level 2 (mediator2). It represents immovable entities. These devices have mainly two tasks. First is, to pass data from layer 1 to CSr and user queries from CSr to lower layers. Second is, to forward data to CSr using the data aggregation scheme for push tags. The second layer's devices take care of the tags present in its region and pass that data to upper layer either at specific intervals or the request arrival, named as the mediator at level 1 (mediator1). At the lowest layer, real-world entities are associated with the tags where each tag maintains a unique node id. These are the items which move most frequently in one's home. We have categorized these tags into two to reduce the database size at the CSr.

- **Push Tags:** These tags broadcast the location of its associated item periodically to the upper layers which are similar to publish/subscribe (pub/sub) mechanism. CSr keeps track of these tags and stores their locations in its local database. So whenever a user queries for one of these items, CSr reply directly from the local database. These tags are associated with the items which are required in the emergency like medical kit, and re-extinguisher and required immediately and frequently misplaced like keys, and wallet.
- **Pull Tags:** These items are relatively more static than the previous ones. These tags will only transmit their location when a user queries for the particular item. Items in this category can be books, clothes, etc., which are not needed immediately and moves less frequently. So whenever a user queries for one of these items, the CSr broadcast query and only one particular tag reply to it.

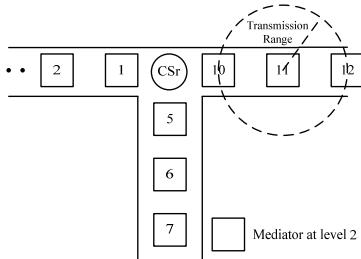
We have kept Arduino boards enabled with wireless capability and connected with the RFID reader module, at various places. The RFID reader reads the tags of the objects in its vicinity and sends it to the Arduino board it is connected with. This Arduino, in turn, sends this tag value to the CSr. The server reads the tag value, maps it to the human understandable object name, and extracts the identity of the sender from where it is coming, location and stores this information in the database.

### 4.1 Data Aggregation Scheme

In a home, there can be the case that some mediator's data cannot reach to CSr directly because it is far away from its range. Like, if we have setup CSr in one of the bedrooms, then it may be possible that mediators of drawing room cannot pass data to CSr directly. Therefore, a data aggregation algorithm is required to aggregate all data in the network at CSr. We proposed data aggregation algorithm for periodically updating the location of push tags at CSr. Algorithm uses the hop-by-hop approach and aggregates the data at CSr from the whole network. There is no need of data aggregation algorithm for pull-based items as they follow the simple request-response approach.

CSr is assigned the lowest id 0. Mediators at layer 2 (mediator2) will also be assigned with the node ids. Mediator2 near to CSr will have less id number than the mediator2 far to CSr. A mediator<sub>2,i</sub> can send data to another mediator<sub>2,j</sub> iff mediator<sub>2,i</sub> node id is less than the node id of mediator<sub>2,j</sub>. E.g., suppose if node id 7 wants to send some data to CSr, then it cannot be direct. Data will be passed to node id 6 and then to node id 5. Finally, node id 5 will pass that data to CSr as shown in Figure 3. Data aggregation algorithm is applied at each mediator2 to collect information about push tags (see Figure 3).

E.g., suppose node id 11 want to send a packet to CSr, then node id 11 will generate a packet and broadcast it. Node id 12 and 10 both will receive it, and they will check the condition, i.e., source id is less than its node id. Node id 10 satisfies this condition so it will broadcast the packet further while node id 12 will discard the received packet (see Figure 3).



**Figure 3. Data aggregation approach**

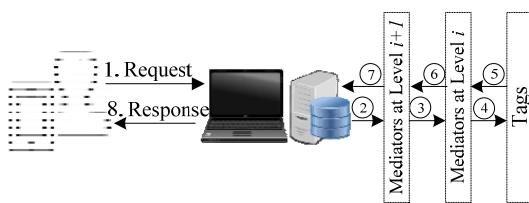
## 4.2 Searching Scheme

Either CSr can store the location of each item and reply whenever a user queries (push-based searching) or search for the item's location in real time whenever a request arrives for it (pull-based searching).



**Figure 4. Sequence of steps for the push-based searching**

In push-based searching approach (pub/sub), each item keeps pushing their details to the upper layer. Tags broadcast their presence periodically which is passed to mediator level 1 to 2 and finally it reaches to CSr using data aggregation algorithm. Once aggregation node does not receive any message from a particular tag, it assumes that item has been moved out of range. This approach guarantees up-to-date information but is not an energy-efficient solution as nodes keep publishing their details periodically. This approach creates huge traffic in the network as well as large database at CSr. On the other hand, maintenance of the large database is cumbersome because of frequent item's location updates. But, this scheme will generate results instantly as CSr will have information about all nodes in the network. Figure 4 shows the sequence of steps required for the push-based searching.



**Figure 5. Sequence of steps for the pull-based searching**

In pull-based searching approach, items/tags do not generate any traffic until user queries for them. In other words, no database is maintained for the items at the CSr while, the location of any item is fetched on user demand. Whenever a request arrives, a request is passed from mediator at level 2 to 1, then mediator at level 1

searches the items in its own region. This technique is energy-efficient as well as network friendly because tags do not have to transmit anything at specified time intervals. But this technique is slower in compared to push-based approach because first of all, a query is broadcasted and user waits till it reaches to all tags and one of them replies. Figure 5. shows the sequence of steps required for the pull-based searching. For the pull-based items, CSr prepares a user request packet ( $I_{pkt}$ ) and broadcast it to all mediators at level 2 which forwards request further up to tags. These tags compare their node id with the requested id, and reply if ids are matched.  $D_{pkt}$  containing the location of the requested item follows the same path in reverse followed by the  $I_{pkt}$ . Once this  $D_{pkt}$  reaches to CSr, it returns the location of sought item to user.

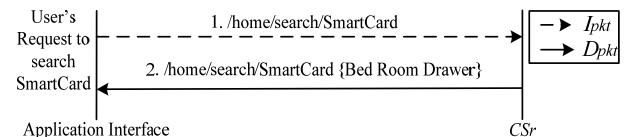
Both schemes have some pros and cons, therefore to maximize the benefits of both schemes; a hybrid scheme, combination of both push and pull-based searching scheme can be used to maintain up-to-date data. Items which are most important and frequently moves, such as keys, wallet, medicines, first-aid kit, etc., can be updated periodically at the CSr. These items will keep pushing their updated location periodically to CSr to reduce the access time. In this scheme, network traffic will also be not very high due to the limited number of updates. Items which are not used frequently and are not required immediately, such as books, clothes, magazines, etc., are considered for the pull-based scheme. To overcome scalability, we are using a hybrid approach, that reduces the network traffic as well as frequent database updates and also delivers quick results for some items. For the problem of real-time updates, the hybrid approach works fine as there will be small number of items which will use push-based approach. Table 1 shows advantages and disadvantages of both approaches.

**Table 1. Comparison between Push and Pull based approach**

Communication Paradigm	Advantages	Disadvantages
Push-based approach	Faster response	High network traffic, Large database
Pull-based approach	Less network traffic	Relatively slow response

## 4.3 Naming scheme

Figure 6 shows  $I_{pkts}$  and  $D_{pkts}$  exchanged to search physical object(s). When a user issues a request to search the smart card, the application interface generates an  $I_{pkt}$   $/home/search/SmartCard$  to CSr. The server searches the requested object in its updated database and returns the corresponding location in human understandable form, such as, Bed Room Drawer.



**Figure 6. Messages exchanged to search object(s) remotely using NDN**

## 4.4 Algorithm for Smart-NDNoT

In this section, we present data aggregation algorithm and searching algorithm for push and pull based items.

## 4.5 Data Aggregation Algorithm

In data aggregation algorithm, each mediator at level 2 waits for the  $I_{pkt}$ . Whenever a packet arrives at mediator<sub>i</sub>, it compares its

node id with the packet's node id. If mediator<sub>i</sub>'s node id is less than the requested one, then that mediator<sub>i</sub> will broadcast the packet further otherwise, packet will be dropped. The same procedure will be repeated by all the mediators between the CSr and mediator<sub>i</sub> (see Figure 7).

#### Data aggregate()

1. Wait for aggregation packet (*Ipkt*)
2. Retrieve source node id of packet received
3. if (source node id > own node id)
4. forward the packet to the next layer (upward)
5. else
6. drop the *Ipkt*

Figure 7. Algorithm for data aggregation

#### 4.5.1 Searching Algorithm

For the ease, the name of the item is mapped to the integer as node id at the CSr. In push-based approach, tags add their node id at the *Ipkt* and broadcast it to the next level mediators using the data aggregation algorithm. When *Ipkt* reaches the CSr, node id is mapped to the item's name and its information is maintained / updated at the database.

#### Push\_tags()

1. Tags add their node id to the *Ipkt*
2. Broadcast the *Ipkt* towards CSr using data aggregation approach
3. Wait for specified period and, go to Step 1

#### Pull\_tags()

1. Wait for incoming *Ipkt* from the user
2. Broadcast *Ipkt* till lower layer
3. if requested-node-id == own node id
4. return the corresponding *Dpkt* to CSr
5. else
6. discard the *Ipkt*

Figure 8. Searching algorithm for Push and Pull based tags

In pull-based approach, a user sends a request to CSr for searching an item using his Smartphone. CSr finds the type of the search and broadcast the *Ipkt* to mediators at the next lower layer. Mediators further broadcast the *Ipkt* till it reaches to the tags. Each tag compares its node id with requested-node-id and returns the corresponding *Dpkt* to CSr, if matched. Otherwise, tag discards the *Ipkt*. Then, CSr returns the result to requested user (see Figure 8).

## 5. IMPLEMENTATION AND TESTING

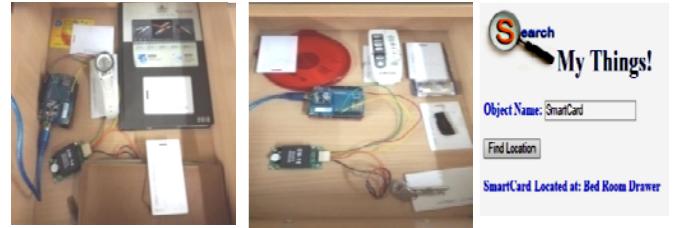
In this section, we discuss the implementation of the NDNoT based object searching system and test it for finding its usability.

In order to strengthen our experimental analysis, we built a prototype with integrated hardware platform and software and based on the prototype develop an NDNoT based object searching mechanism for the smart home to demonstrate the usability of the *Search-NDNoT* framework.

### 5.1 Testbed Setup

In our system, CSr comprises of a laptop, an Arduino (a programmable microcontroller board based on ATMEGA328), XBee Radio module (802.15.4 protocol), XBee wireless shield, XBee Explorer, etc. The messages are stored on the 2 GB SD card attached to the Arduino wireless SD shield. For the implementation purpose, we have eliminated mediator1, i.e., sensors are directly connected to mediator2 because transmission

range of mediators should not overlap. Mediator2 is implemented using the Arduino, XBee wireless shield, XBee Radio module (802.15.4 protocol) and RFID Reader. Moreover, items are attached with the RFID tags (see Figure 9 (a) and (b)). Sensors periodically update their values and send to the microcontroller. A sketch on Arduino is written using C/C++ which is customizable according to the hardware characteristics.



(a) Bed Room Drawer      (b) Drawing Room Drawer      (c) Interface for Object Access

Figure 9. Searching objects in the smart home using NDN

The user launches the application "Search My Things" on his Smartphone, and types the name of the object he wants to search, Figure 9 (c). The application then displays the approximate location of the object in human understandable form like "SmartCard Located at: Bed Room Drawer".

## 5.2 Experiment Result

In this section, we discuss the performance comparison of the proposed *Search-NDNoT* system with respect to pull and push-based communication. We have carried out experiments on a prototype testbed system. To evaluate the performance of our proposed scheme, we have randomly searched 10 items in the closed vicinity for both push and pull based search scheme and we have chosen Round Trip Time (RTT) as the performance metric. Below, we have formally defined the performance metric.

- Round Trip Time (RTT): RTT is the total time elapsed between a user sends a request and receives a corresponding reply.

First, we started with only two numbers of requests and repeated the procedure for ten times to mitigate the effect of hardware dependency. We increased the number of requests one-by-one to analyze the impact of the number of requests on the network. Then, an average is taken.

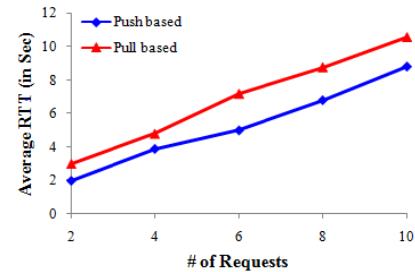


Figure 10. Average RTT for pull and push based search scheme

Figure 10 shows that the average RTT for pull and push based search scheme which is almost constant with increasing number of requests.

## 6. CONCLUSION AND FUTURE WORKS

Object search in the smart home is an important application of the IoT. But, current Internet is facing some challenges. To overcome

from the Internet shortcomings, researchers proposed NDN, a future Internet networking paradigm as a solution. In this paper, we proposed an NDN-based object searching mechanism for the smart home. We have developed a complete prototype of the system. Moreover, extensive experiments are done on prototype testbed to validate the performance of NDN. In future, we will extend the usability of our proposed system for the different scenarios and applications of the smart home.

## 7. ACKNOWLEDGEMENT

This work is partially supported by the Alexander von Humboldt Foundation through the post-doctoral research fellow Dr. Vaskar Raychoudhury.

## 8. REFERENCES

- [1] Shang, Wentao, et al. "Named Data Networking of Things." *IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 2016.
- [2] M. Wang, J. Cao, Y. Sun, J. Li, "Toward Ubiquitous Searching", *13th IEEE International Conference on Parallel and Distributed Systems*, Dec. 5 – 7, 2007, pp. 1-8.
- [3] M. Chen, et al., "An efficient tag search protocol in large-scale RFID systems," *IEEE International conference of INFOCOM 2013*, pp. 899-907.
- [4] Wang H. and Tan C.C. and Li Q., "Snoogle: A Search Engine for Pervasive Environments", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 21, No. 8, pp.1188-1202, August 2010.
- [5] Tan C.C. and Sheng B. and Wang H. and Li Q., "Microsearch: A Search Engine for Embedded Devices Used in Pervasive Computing", *ACM Transactions on Embedded Computing Systems (TECS)*, Vol. 9, No. 4, pp. 1-29, New York, March 2010.
- [6] D. Saxena, et al., "Named Data Networking: A Survey," *Elsevier Computer Science Review*, 2016.
- [7] L. Zhang, et al., (2010). Named Data Networking (NDN) Project. [Online]. Available: <http://named-data.net/project/annual-progress-summaries/> [Accessed 10-Sep-2015].
- [8] D. Saxena and V. Raychoudhury, "Radient: Scalable, Memory Efficient Name Lookup Algorithm for Named Data Networking", *Elsevier Journal of Network and Computer Applications*, Volume 63, Jan., 2016.
- [9] D. Saxena and V. Raychoudhury, "N-FIB: Scalable, Memory Efficient Name-based forwarding", *Elsevier Journal of Network and Computer Applications*, September, 2016.
- [10] D. Saxena and V. Raychoudhury, C. Becker and N. Suri. "Reliable Memory Efficient Name Forwarding in Named Data Networking", *IEEE EUC*, 2016, Paris, France.
- [11] J. Burke, et al., "Authenticated lighting control using named data networking." UCLA, NDN Technical Report NDN-0011(2012).
- [12] A. Ghodsi, et al., "Information-centric networking: Ready for the real world?" (Seminar12361), vol. 2, no. 9, pp. 114, 2012.
- [13] E. Baccelli, et al., "Information centric networking in the IoT: experiments with NDN in the wild." arXiv: 1406.6608 (2014).
- [14] J. Zhang, Q. Li, et al., "iHEMS: an information-centric approach to secure home energy management," *IEEE International Conference of SmartGrid Comm.*, 2012, pp. 217-222.
- [15] R. Ravindran, et al., "Information-centric networking based homenet," *IFIP/IEEE International Symposium on Integrated Network Management (IM) 2013*, pp. 1102-1108.
- [16] J. Burke, and L. Zhang, "Securing Building Management Systems Using Named Data Networking," *IEEE Network*, vol. 28, no. 3, pp. 50–56, May-June 2014.
- [17] J. Burke, P. Gasti, N. Nathan, and G. Tsudik, "Securing Instrumented Environments over Content-Centric Networking: the Case of Lighting Control and NDN," *IEEE International Workshop on INFOCOM-NOMEN 2013*, pp. 394-398.
- [18] Y. Zhang, et al., "ICN based Architecture for IoT," in Internet-Draft, June 2014.
- [19] M. Amadeo, et al., "Named Data Networking for IoT: an Architectural Perspective," *IEEE European Conference on Networks and Communications (EuCNC)*, Bologna, Italy, 2014, pp. 1-5.
- [20] J. Francois, et al., "CCN Traffic Optimization for IoT," *4th International Conf. on Network of the Future*, 2013, pp. 1-5
- [21] D. Saxena, et al., "SmartHealth-NDNoT: Named Data Network of Things for Healthcare Services," *ACM International Workshop on Pervasive Wireless Healthcare (MobiHoc-MobileHealth)*, 2015, pp. 45-50.

# Efficient Algorithms for Power Maximization in the Vector Model for Wireless Energy Transfer

Ioannis Katsidimas

Department of Computer  
Engineering and Informatics,  
University of Patras, Greece  
Computer Technology Institute  
and Press “Diophantus”,  
Greece

[ikatsidima@ceid.upatras.gr](mailto:ikatsidima@ceid.upatras.gr)

Sotiris Nikoletseas

Department of Computer  
Engineering and Informatics,  
University of Patras, Greece  
Computer Technology Institute  
and Press “Diophantus”,  
Greece

[nikole@cti.gr](mailto:nikole@cti.gr)

Theofanis P. Raptis

Institute for Informatics and  
Telematics, National Research  
Council, Italy  
Department of Computer  
Engineering and Informatics,  
University of Patras, Greece

[theofanis.raptis@iit.cnr.it](mailto:theofanis.raptis@iit.cnr.it)

Christoforos Raptopoulos

Department of Computer  
Engineering and Informatics,  
University of Patras, Greece  
Computer Technology Institute  
and Press “Diophantus”,  
Greece

[raptopox@ceid.upatras.gr](mailto:raptopox@ceid.upatras.gr)

## ABSTRACT

Rapid technological advances in the domain of Wireless Power Transfer (WPT) pave the way for novel methods for power management in systems of wireless devices and recent research works have already started considering algorithmic solutions for tackling emerging problems. However, those works are limited by the system modelling, and more specifically *the one-dimensional abstraction* suggested by Friis formula for the power received by one antenna under idealized conditions given another antenna some distance away.

Different to those works, we *use a model* which arises naturally from fundamental properties of the *superposition of energy fields*. This model has been shown to be more realistic than other one-dimensional models that have been used in the past and can capture superadditive and cancellation effects. Under this model, we define *two new interesting problems* for configuring the wireless power transmitters so as to maximize the total power in the system and we prove that the first problem can be solved in polynomial time. We present a *distributed solution* that runs in pseudo-polynomial time and uses various knowledge levels and we provide theoretical performance guarantees. Finally, we design *three heuristics* for the second problem and evaluate them experimentally.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

ICDCN '17, January 04-07, 2017, Hyderabad, India

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007749>

## CCS Concepts

- Computing methodologies → *Distributed algorithms*;
- Power and Energy → *Power networks; Power optimization*;

## Keywords

Wireless Power Transfer; Power Management; Algorithms; Models

## 1. INTRODUCTION

Wireless Power Transfer (WPT) has recently become a commercially viable option in various wireless systems due to the reliability of continuous power supply and the convenience provided by the fact that no static (wired) network connections are needed between the devices. The efficiency of the various technological alternatives is increasing every year. Current fast-charging protocols achieve up to 84% efficiency for wireless power transfer up to distances of 15 meters [1], while at the same time keeping thermal dissipation significantly low [2]. A WPT enabled system consists of several wireless transmitter and receiver devices. A *wireless transmitter (charger)* is a device that has a dedicated power source with significant power supply and can transfer power wirelessly to receivers. A *receiver (node)* is a device that is powered by harvesting the radio frequency energy from the chargers. A receiver is usually an electronic device that is needed to perform a specific task in the wireless system, for example a sensor mote in a wireless sensor network. Systems of wireless devices have to operate under increasing demands of power in order to sustain various computational and communication tasks. For this reason, the efficient and distributed cooperation of the transmitters and receivers towards achieving an effective power allocation in the system is a crucial task.

**Our contribution.** While considerable research efforts

have been invested into power management in wireless systems, most of the models studied in the literature neglect key features of electromagnetic fields. As a result, such models are unable to explain various phenomena occurring in real applications (e.g., cancellation and superadditive charging effects; see Section 3), yielding some algorithmic solutions impractical. In view of the above, our contribution in this paper is the following:

- We algorithmically study a more realistic model for WPT in wireless systems, which was nicely initiated in [12]. In particular, this is a “vector” model that takes into account the *superposition* of electromagnetic fields created by independent wireless transmitters, as well as fundamental properties of the superposition of waves from physics. To the best of our knowledge, this is the first algorithmic study of a vector model for WPT.
- We define *two new computational problems* for the efficient utilization of power resources in a wireless system consisting of a family of transmitters  $\mathcal{C}$  and a family of receivers  $\mathcal{R}$ . In particular, we first consider the problem **MAX-POWER** of finding a configuration (set of operation levels) for transmitters that *maximizes the total power* received by  $\mathcal{R}$ . Second, we consider the problem **MAX- $k$ MIN-GUARANTEE** of finding a configuration that maximizes the *minimum cumulative power among all the sets of  $k$  receivers*.
- We formulate **MAX-POWER** as a *quadratic program* and we prove that we can find an optimal solution efficiently by presenting a *family of distributed algorithms* using different levels of knowledge of the system. We prove that these algorithms run in pseudopolynomial time, but we experimentally show that they are quite faster in practice.
- We design and experimentally evaluate *three efficient heuristics* for **MAX- $k$ MIN-GUARANTEE** that provide good approximations of the optimal solution. The first heuristic is a generalization of our algorithmic solution to **MAX-POWER**, while the second samples a few representative  $k$ -sets of receivers and then solves the problem considering only those sets. Finally, our third heuristic is a hybrid of the previous two ideas and we show that it outperforms both in typical deployments of transmitters and receivers on the plane.

## 2. RELATED WORK

Wireless Power Transfer methods in large scale networked systems have attracted much attention from researchers worldwide. Several works study applications in sensor networks [3, 7, 16] and wireless distributed systems [10, 15], UAVs [6, 11]. Different to all those works, in this paper we investigate a static setting, where transmitter and receiver locations are predefined and stationary, and there are no mobile elements in the system.

There have been some works on closely related themes to this paper. In [17], given a set of candidate locations for placing chargers, the authors provide a charger placement and a corresponding power allocation to maximize the charging quality, subject to a power budget. In a recent paper [14], a subset of the authors of this paper study the Low Radiation Efficient Charging Problem, in which we optimize the

amount of useful energy transferred from chargers to nodes (under constraints on the maximum level of imposed electromagnetic radiation). In a similar setting, the authors in [5] consider the Safe Charging with Adjustable Power (SCAPE) problem for adjusting the power of chargers to maximize the charging utility of devices, while assuring that electromagnetic intensity at any location in the field does not exceed a given threshold. It is worth noting that, even though all the above works nicely demonstrate the gains of carefully distributing the power in a wireless setting, they use one-dimensional models, which fall short of capturing various intricate aspects of WPT.

## 3. THE CHARGING MODEL

In a recent paper ([12]), the authors considered a model for the superposition of electromagnetic fields created by independent wireless energy sources, which takes into account fundamental properties of the superposition of waves from physics. The model of [12] goes beyond (in fact, it is a generalization of) the one-dimensional abstraction suggested by Friis’ formula for the power received by one antenna under idealized conditions given another antenna some distance away. In particular, the *electric field* created by an energy transmitter (charger)  $C$ , operating at full capacity, at a receiver  $R$  at distance  $d = \text{dist}(C, R)$  is a 2-dimensional vector given by

$$\mathbf{E}(C, R) \stackrel{\text{def}}{=} \beta \cdot \frac{1}{d} \cdot e^{-j\frac{2\pi}{\lambda}d} = \beta \cdot \frac{1}{d} \cdot \begin{bmatrix} \cos\left(\frac{2\pi}{\lambda}d\right) \\ \sin\left(\frac{2\pi}{\lambda}d\right) \end{bmatrix}, \quad (1)$$

where  $\lambda$  depends on the frequency at which the transmitter operates, and  $\beta$  is a constant that depends on the hardware of the transmitter and the environment.<sup>1</sup>

The main point of the charging model of [12], which also sets it apart from other (less realistic, but more tractable) models in the wireless charging literature, is that the total electric field created by a family of energy transmitters  $\mathcal{C}$  at a receiver  $R$  is the *superposition (vector-sum)* of their individual electric fields, that is

$$\mathbf{E}(\mathcal{C}, R) \stackrel{\text{def}}{=} \sum_{C \in \mathcal{C}} \mathbf{E}(C, R). \quad (2)$$

Furthermore, the total available *power* at the receiver  $R$  is given by

$$P(\mathcal{C}, R) = \gamma \cdot \|\mathbf{E}(\mathcal{C}, R)\|^2, \quad (3)$$

where  $\|\cdot\|$  denotes the length (2-norm) of the vector. The constant  $\gamma$  depends on the hardware of the transmitter, the hardware of the receiver and the RF-to-DC conversion efficiency.

It is worth noting that the above model arises naturally from fundamental properties of the superposition of energy fields and has been shown to be more realistic than other one-dimensional models that have been used in the past and can capture *superadditive* and *cancellation* effects [8,

<sup>1</sup>In fact, the exact formula used in [12] for the electric field is  $\mathbf{E}(C, R) \stackrel{\text{def}}{=} \sqrt{\frac{Z_0 G_C P_C}{4\pi d^2}} \cdot e^{-j\frac{2\pi}{\lambda}d}$ , where  $Z_0$  is a physical constant indicating the wave-impedance of a plane wave in free space,  $G_C$  is the gain and  $P_C$  is the output power of the transmitter. In this paper, without loss of generality of our algorithmic solutions, we assume that all wireless transmitters and receivers are identical, thus the aforementioned parameters are the same for each charger.

9, 13]. To fix ideas and to demystify the above definitions, we present the following fictitious example: Assume there are two transmitters  $C_1$  and  $C_2$  placed at points  $(0, 0)$  and  $(2, 0)$  in the 2-dimensional plane. First, consider a receiver  $R$  placed at  $(1, 0)$ . Assume also, for the sake of clarity that all constants in the above model are set to 1, i.e.  $\lambda = \beta = \gamma = 1$ . When only one of the two transmitters is operational, the power received by  $R$  is  $P(C_1, R) = P(C_2, R) = \|\mathbf{E}(C_1, R)\|^2 = \|\mathbf{E}(C_2, R)\|^2 = \left(\frac{1}{\text{dist}(C_1, R)}\right)^2 = 1$ . On the other hand, if both transmitters are operational, the power received by  $R$  is given by equation (2), that is

$$P(\{C_1, C_2\}, R) = \|\mathbf{E}(C_1, R) + \mathbf{E}(C_2, R)\|^2.$$

Furthermore, it is not hard to see that, since  $R$  is equidistant from either  $C_1$  or  $C_2$ , the vectors  $\mathbf{E}(C_1, R)$  and  $\mathbf{E}(C_2, R)$  point to the same direction. Therefore,  $P(\{C_1, C_2\}, R) = 4P(C_1, R) = 2(P(C_1, R) + P(C_2, R)) = 4$ . Notice then that the power received by  $R$  when both transmitters are operational is larger than the sum of the powers it receives when only one of the transmitters is operational; this is the so-called superadditive effect and is visible in local maxima in the curve shown in Figure 1b.

Second, consider a receiver  $R'$  placed at  $(\frac{5}{4}, 0)$ . Then by equation (1),  $\mathbf{E}(C_1, R') = \frac{4}{5} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ , and also  $\mathbf{E}(C_2, R') = \frac{4}{3} \cdot \begin{bmatrix} 0 \\ -1 \end{bmatrix}$ . By equation (2), the power received by  $R'$  when both transmitters are operational is  $P(\{C_1, C_2\}, R') = (\frac{8}{15})^2 \approx 0.28$ . Notice then that the power received by  $R'$  when both transmitters are operational is much less than  $\min\{P(C_1, R'), P(C_2, R')\} = (\frac{4}{5})^2 \approx 0.64$ ; this is the so-called cancellation effect and is visible in local minima in the curve shown in Figure 1b.

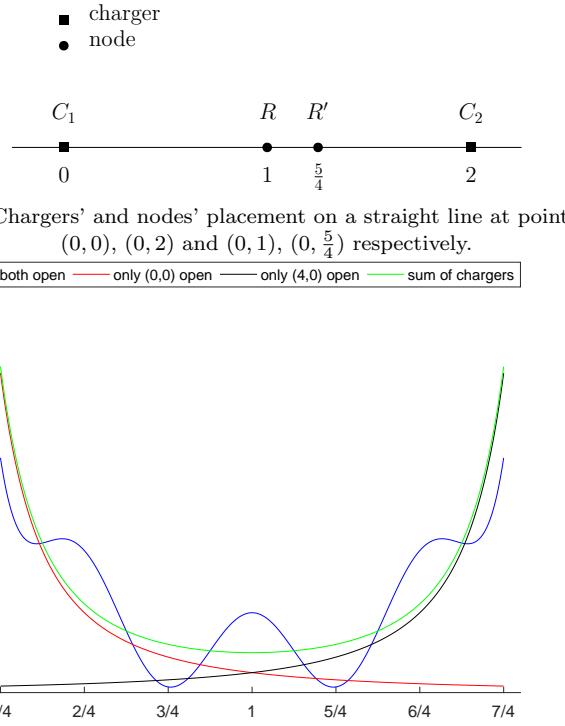
In view of our discussion in this paper, it is worth noting that, even in the above toy example, it is non-trivial to provide a closed formula for the point in the line between the two chargers where the received power is maximized.

### 3.1 A basic assumption

A necessary condition that is required in Friis' formula, and by extension in the above model, is that the wireless transmitter and receiver need to be at distance at least  $\lambda$ . In fact, for transmitters to receivers distances less than  $\lambda$ , more complex laws apply but we do not consider them here as they are beyond the algorithmic focus of the paper. A similar constraint also holds for the power received by receivers that are very close to each other. In particular, if receivers  $R, R'$  are closer than  $\frac{\lambda}{2\pi}$  apart, then the power received by each receiver no longer follows Friis' law. This was to be expected, since otherwise we could have wireless transmitters of bounded capabilities that could theoretically provide infinite power (e.g. by placing receivers arbitrarily close to each other and to the transmitter).

In this paper, we consider algorithmic problems related to power guarantees with respect to a fixed deployment of a family  $\mathcal{C}$  of wireless transmitters and a family  $\mathcal{R}$  of receivers. To avoid confusion, we will assume that any placement of chargers and receivers satisfies the above *placements constraints*. In particular: (a) For each charger  $C \in \mathcal{C}$  and receiver  $R \in \mathcal{R}$ , we have  $\text{dist}(C, R) \geq \lambda$  and (b) for any pair of receivers  $R, R' \in \mathcal{R}$ , we have  $\text{dist}(R, R') \geq \frac{\lambda}{2\pi}$ .

We finally note that, since  $\frac{\lambda}{2\pi}$  is usually smaller than a



(b) The power distribution between the two chargers. Different curves represent different operation levels of the chargers.

Figure 1: Example showing the superadditive and cancellation effects.

few centimeters, in practical situations, the above placement constraints will not be restrictive, as the non-trivial volume of any transmitting or receiving device guarantees that transmitters and chargers are far enough from each other.

## 4. PROBLEM DEFINITION

Consider a system consisting of a family  $\mathcal{C}$  of identical wireless chargers and a family  $\mathcal{R}$  of identical wireless receivers (nodes). For each charger  $C \in \mathcal{C}$ , we denote by  $\mathbf{x}_C \in [0, 1]$  a variable that determines the level of operation of  $C$ . In particular, generalizing equation (1), the electric field vector created by  $C$  at the location of a receiver  $R$ , when the former operates at level  $\mathbf{x}_C$  is given by  $\mathbf{x}_C \cdot \mathbf{E}(C, R) = \mathbf{x}_C \cdot \beta \cdot \frac{1}{\text{dist}(C, R)} \cdot e^{-j \frac{2\pi}{\lambda} \text{dist}(C, R)}$ .

We will refer to the vector  $\mathbf{x} \in [0, 1]^{\mathcal{C}}$  as the *configuration* of the chargers in the system. Slightly abusing notation, we will denote by  $\mathcal{C}(\mathbf{x})$  a family of chargers that operate according to configuration  $\mathbf{x}$ .

We initially consider the following problem:

**DEFINITION 1 (MAX-POWER).** *Given a family of chargers  $\mathcal{C}$  and family of receivers  $\mathcal{R}$  that satisfy the placement constraints of Subsection 3.1, find a configuration for the chargers that maximizes the total power to  $\mathcal{R}$ . That is, find  $\mathbf{x}^*$  such that*

$$\mathbf{x}^* \in \arg \max_{\mathbf{x} \in [0, 1]^{\mathcal{C}}} P(\mathcal{C}(\mathbf{x}), \mathcal{R}), \quad (4)$$

where  $P(\mathcal{C}(\mathbf{x}), \mathcal{R}) = \sum_{R \in \mathcal{R}} P(\mathcal{C}(\mathbf{x}), R)$ .

We will denote by  $\binom{\mathcal{R}}{k}$  the family of all subsets of  $\mathcal{R}$  containing  $k$  nodes. In this paper, we also study the following generalization of MAX-POWER, which finds a configuration that provides a minimum charging guarantee among all  $k$ -sets of nodes:

**DEFINITION 2 (MAX- $k$ MIN-GUARANTEE).** *Given a family of chargers  $\mathcal{C}$  and a family of receivers  $\mathcal{R}$  that satisfy the placement constraints of Subsection 3.1, find a configuration for the chargers that maximizes the minimum cumulative power among all subsets of  $\mathcal{R}$  of size  $k$ . That is, find  $\mathbf{x}^*$  such that*

$$\mathbf{x}^* \in \arg \max_{\mathbf{x} \in [0,1]^m} \min_{A \in \binom{\mathcal{R}}{k}} P(\mathcal{C}(\mathbf{x}), A), \quad (5)$$

where  $P(\mathcal{C}(\mathbf{x}), A) = \sum_{R \in A} P(\mathcal{C}(\mathbf{x}), R)$ .

## 5. MAXIMUM TOTAL POWER

In this section we present an efficient algorithm for MAX-POWER. For simplicity, consider a family of wireless chargers  $\mathcal{C} = \{C_1, \dots, C_m\}$ , where  $m = |\mathcal{C}|$ , and a family of receivers  $\mathcal{R} = \{R_1, \dots, R_n\}$ , where  $n = |\mathcal{R}|$ . Let also  $\mathbf{x} \in [0,1]^m$  be the configuration of the chargers, where  $\mathbf{x}_j$  is the level of operation of charger  $C_j$ ,  $j \in [m]$ .

We first show that the MAX-POWER problem can be expressed as a quadratic program. To this end, for each  $R \in \mathcal{R}$ , define  $\mathbf{Q}^{(R)}$  be a  $2 \times m$  matrix whose  $j$ -th column is the 2-dimensional vector of the electric field created from  $C_j$  at  $R$ , i.e.  $\mathbf{Q}_{:,j}^{(R)} = \sqrt{\gamma} \cdot \mathbf{E}(C_j, R)$ , for each  $j \in [m]$ . Notice now that we can write the power harvested by the receiver  $R$  as follows:

$$\begin{aligned} P(\mathcal{C}(\mathbf{x}), R) &= \gamma \|\mathbf{E}(\mathcal{C}(\mathbf{x}), R)\|^2 \\ &= \gamma \left\| \sum_{C \in \mathcal{C}} \mathbf{x}_C \mathbf{E}(C, R) \right\|^2 \\ &= \left( \sum_{C \in \mathcal{C}} \mathbf{x}_C \sqrt{\gamma} \mathbf{E}(C, R) \right)^T \left( \sum_{C \in \mathcal{C}} \mathbf{x}_C \sqrt{\gamma} \mathbf{E}(C, R) \right) \\ &= (\mathbf{Q}^{(R)} \mathbf{x})^T \mathbf{Q}^{(R)} \mathbf{x}, \end{aligned}$$

where  $(\cdot)^T$  denotes the transpose of a matrix or vector. Therefore, setting  $\mathbf{H} \stackrel{\text{def}}{=} \sum_{R \in \mathcal{R}} (\mathbf{Q}^{(R)})^T \mathbf{Q}^{(R)}$ , the solution to MAX-POWER is given by

$$\mathbf{x}^* \in \arg \max_{\mathbf{x} \in [0,1]^m} \mathbf{x}^T \mathbf{H} \mathbf{x}.$$

It is worth noting that, in general, the maximization of a quadratic form is a non-convex quadratic program (even when  $\mathbf{H}$  is positive semi-definite, which is the case here), hence cannot be solved in polynomial time. Nevertheless, by taking into account several properties and the special form of our problem, we are able to provide an efficient algorithm for MAX-POWER.

We first need the following elementary lemma that considerably reduces the size of the search space.

**LEMMA 1.** *If  $\mathbf{x}^*$  is an optimal solution to MAX-POWER, then  $\mathbf{x}^* \in \{0,1\}^m$ . In particular, there exists an optimal solution to MAX-POWER in which each charger either operates at full capacity or not at all.*

**PROOF.** Let  $\mathbf{y} \in [0,1]^m$  be such that

(a)  $\mathbf{y} \in \arg \max_{\mathbf{x} \in [0,1]^m} \mathbf{x}^T \mathbf{H} \mathbf{x}$  (i.e. it is an optimal solution to MAX-POWER) and (b)  $\mathbf{y}$  has the minimum number of elements that are neither 0 nor 1; let also  $k$  be the number of such elements, i.e.  $k \stackrel{\text{def}}{=} |\{i : \mathbf{y}_i \notin \{0,1\}\}|$ . If  $k = 0$ , then there is nothing to prove, so we will assume for the sake of contradiction that  $k > 0$ . Let  $j$  be an index such that  $\mathbf{y}_j \notin \{0,1\}$  and define vectors  $\mathbf{y}^{(0)}$  and  $\mathbf{y}^{(1)}$  that are equal to  $\mathbf{y}$  everywhere, except on position  $j$ , where  $\mathbf{y}_j^{(0)} = 0$  and  $\mathbf{y}_j^{(1)} = 1$ . Notice now that, by the optimality and minimality assumptions on  $\mathbf{y}$ , we have that  $\mathbf{y}^T \mathbf{H} \mathbf{y} > (\mathbf{y}^{(0)})^T \mathbf{H} \mathbf{y}^{(0)}$  and  $\mathbf{y}^T \mathbf{H} \mathbf{y} > (\mathbf{y}^{(1)})^T \mathbf{H} \mathbf{y}^{(1)}$ .

On the other hand, for  $z \in (0,1)$ , define the vector  $\mathbf{y}^{(z)} \stackrel{\text{def}}{=} (1-z)\mathbf{y}^{(0)} + z\mathbf{y}^{(1)}$ . In particular,  $\mathbf{y}^{(z)}$  is the  $m$ -dimensional vector that is equal to  $\mathbf{y}$  everywhere, except on position  $j$ , where  $\mathbf{y}_j^{(z)} = z$ . Consider now the function  $f(z) = (\mathbf{y}^{(z)})^T \mathbf{H} \mathbf{y}^{(z)}$ . Notice that  $f(z)$  is a single variable polynomial of degree 2, and it is a simple matter of calculus to show that its second derivative satisfies  $\frac{d^2 f}{dz^2} = \mathbf{H}_{j,j} = \sum_{R \in \mathcal{R}} \gamma \|\mathbf{E}(C_j, R)\|^2 = P(C_j, \mathcal{R})$ , which is strictly positive. But this implies that  $f(z) < \max\{f(0), f(1)\}$ , for all  $z \in (0,1)$ . However, this is a contradiction, since we have already established that, by assumption,  $f(\mathbf{y}_j) = \mathbf{y}^T \mathbf{H} \mathbf{y} > \max\{(\mathbf{y}^{(0)})^T \mathbf{H} \mathbf{y}^{(0)}, (\mathbf{y}^{(1)})^T \mathbf{H} \mathbf{y}^{(1)}\} = \max\{f(0), f(1)\}$ . We conclude that in any optimal solution to MAX-POWER each charger either operates at full capacity or not at all.  $\square$

We now prove a useful property of global maxima of the objective function  $\mathbf{x}^T \mathbf{H} \mathbf{x}$  in  $[0,1]^m$ . The proof uses properties of positive semi-definite (PSD) matrices (see [4] for an introduction to PSD matrices and their properties). We note here that, Lemma 1 and Theorem 1 below imply that any local maxima of the objective function  $P(\mathcal{C}(\mathbf{x}), R) = (\mathbf{Q} \mathbf{x})^T \mathbf{Q} \mathbf{x}$  are also global maxima that belong to  $\{0,1\}^m$ . In particular, this means that the gradient descent method can be used to find a global maximum (i.e. an optimal solution to MAX-POWER). Nevertheless, in our experimental evaluation, we used a pseudopolynomial distributed algorithm for computing the *exact* optimum configuration for MAX-POWER, which is quite fast in practice. We present this algorithm later in this section.

**THEOREM 1.** *A configuration  $\mathbf{x}^* \in \{0,1\}^m$  is an optimal solution to MAX-POWER if and only if  $P(\mathcal{C}(\mathbf{x}^*), \mathcal{R}) \geq P(\mathcal{C}(\mathbf{y}), \mathcal{R})$ , for each  $\mathbf{y}$  that comes from  $\mathbf{x}$  by setting exactly one of its coordinates to either 0 or 1.*

**PROOF.** For a configuration  $\mathbf{x} \in \{0,1\}^m$  and for all  $j \in [m]$  and  $a \in \{0,1\}$ , define  $\mathbf{x}^{(j,a)} \stackrel{\text{def}}{=} \mathbf{x} + (a - \mathbf{x}_j) \mathbf{e}_j$ , where  $\mathbf{e}_j$  is the  $j$ -th vector in the standard basis of  $\mathbb{R}^m$ . Notice that  $\mathbf{x}^{(j,1)}$  (respectively  $\mathbf{x}^{(j,0)}$ ) is the configuration that is identical to  $\mathbf{x}$ , with the only difference that charger  $j$  operates at full capacity (respectively does not operate).

Clearly, if  $\mathbf{x}^*$  is an optimal solution, then  $P(\mathcal{C}(\mathbf{x}^*), \mathcal{R}) \geq P(\mathcal{C}(\mathbf{x}^{*(j,a)}), \mathcal{R})$ , for any  $j \in [m]$  and  $a \in \{0,1\}$ . Therefore, it remains to prove the “only if” part of the Theorem. To this end, let  $\mathbf{x}^*$  be such that  $P(\mathcal{C}(\mathbf{x}^*), \mathcal{R}) \geq P(\mathcal{C}(\mathbf{x}^{*(j,a)}), \mathcal{R})$ , for any  $j \in [m]$  and  $a \in \{0,1\}$ , and assume for the sake

of contradiction that there is a configuration  $\mathbf{z}$  such that  $P(\mathcal{C}(\mathbf{x}^*), \mathcal{R}) < P(\mathcal{C}(\mathbf{z}), \mathcal{R})$ .

By Lemma 1, we only need to consider configurations in  $\{0, 1\}^m$ . Therefore, assume that  $\mathbf{x}^* \in \{0, 1\}^m$  and  $\mathbf{z} = \mathbf{x}^* + \sum_{j=1}^m a_j \mathbf{e}_j = \mathbf{x}^* + \mathbf{a}$ , for some  $\mathbf{a} \in \{-1, 0, 1\}^m$ , such that  $\mathbf{z}^T \mathbf{H} \mathbf{z} = P(\mathcal{C}(\mathbf{z}), \mathcal{R}) > P(\mathcal{C}(\mathbf{x}), \mathcal{R}) = \mathbf{x}^{*T} \mathbf{H} \mathbf{x}^*$ .

Note that, since  $\mathbf{H}$  is symmetric, for any  $j \in [m]$  we have  $\mathbf{e}_j^T \mathbf{H} \mathbf{x}^* = \mathbf{x}^{*T} \mathbf{H} \mathbf{e}_j$ , and so

$$(\mathbf{x}^* + a_j \mathbf{e}_j)^T \mathbf{H} (\mathbf{x}^* + a_j \mathbf{e}_j) = \mathbf{x}^{*T} \mathbf{H} \mathbf{x}^* + 2a_j \mathbf{e}_j^T \mathbf{H} \mathbf{x}^* + a_j^2 \mathbf{H}_{j,j}.$$

Rearranging, and using the assumption that  $P(\mathcal{C}(\mathbf{x}^*), \mathcal{R}) \geq P(\mathcal{C}(\mathbf{x}^* + a_j \mathbf{e}_j), \mathcal{R})$ , we get

$$2a_j \mathbf{e}_j^T \mathbf{H} \mathbf{x}^* + a_j^2 \mathbf{H}_{j,j} \leq 0, \quad (6)$$

for any  $j \in [m]$ . By a similar computation, we have

$$\begin{aligned} \mathbf{z}^T \mathbf{H} \mathbf{z} &= \mathbf{x}^{*T} \mathbf{H} \mathbf{x}^* + 2\mathbf{a}^T \mathbf{H} \mathbf{x}^* + \mathbf{a}^T \mathbf{H} \mathbf{a} \\ &= \mathbf{x}^{*T} \mathbf{H} \mathbf{x}^* + \sum_{j=1}^m 2a_j \mathbf{e}_j^T \mathbf{H} \mathbf{x}^* + \mathbf{a}^T \mathbf{H} \mathbf{a}. \end{aligned} \quad (7)$$

Summing (6) over all  $j \in [m]$  and substituting in (7), we get

$$\mathbf{z}^T \mathbf{H} \mathbf{z} - \mathbf{x}^{*T} \mathbf{H} \mathbf{x}^* \leq - \sum_{j=1}^m a_j^2 \mathbf{H}_{j,j} + \mathbf{a}^T \mathbf{H} \mathbf{a}. \quad (8)$$

It is now a simple matter of algebra to show that the right hand side of the above inequality is non-positive. Indeed, let  $\mathbf{H}'$  be the  $m' \times m'$  principal submatrix of  $\mathbf{H}$  corresponding to rows (and columns)  $j$  for which  $a_j \neq 0$  (in particular,  $m'$  is the number of non-zero elements of  $\mathbf{a}$ ). Clearly, since  $\mathbf{H}$  is positive semi-definite, then so is  $\mathbf{H}'$ . Let  $\lambda'_1 \geq \dots \geq \lambda'_{m'} \geq 0$  and  $\mathbf{v}'_1, \dots, \mathbf{v}'_{m'}$  be the eigenvalues and eigenvectors of  $\mathbf{H}'$ . Now notice that  $\sum_{j=1}^m a_j^2 \mathbf{H}_{j,j} = \sum_{j=1}^{m'} a_j^2 \mathbf{H}'_{j,j} = \text{tr}(\mathbf{H}') = \sum_{j=1}^{m'} \lambda'_j$ , where  $\text{tr}(\mathbf{H}')$  is the trace of  $\mathbf{H}'$  and we have used the fact that the trace of a matrix is equal to the sum of its eigenvalues. Finally, since  $\mathbf{a}$  is an orthonormal rotation of the vector  $\sum_{j=1}^{m'} \mathbf{v}'_j$ , we have  $\mathbf{a}^T \mathbf{H} \mathbf{a} = \sum_{j=1}^{m'} \lambda'_j$ .

In view of the above, by inequality (8), we get that  $\mathbf{z}^T \mathbf{H} \mathbf{z} - \mathbf{x}^{*T} \mathbf{H} \mathbf{x}^* \leq 0$ , which is a contradiction. Therefore, we conclude that if  $P(\mathcal{C}(\mathbf{x}^*), \mathcal{R}) \geq P(\mathcal{C}(\mathbf{x}^{*(j,a)}), \mathcal{R})$ , for any  $j \in [m]$  and  $a \in \{0, 1\}$ , then  $\mathbf{x}^*$  is an optimal solution.  $\square$

Lemma 1 and Theorem 1 suggest that the following distributed algorithm (which we call **IterativeMaxPower**) can be used to find an *exact* optimum configuration for **MAX-POWER**: Initially, we begin from an arbitrary configuration in  $\{0, 1\}^m$ . In each subsequent step, we parse the set of chargers in order to find a charger  $C \in \mathcal{C}$  such that the total power received by  $R$  can be increased by flipping the operation level of  $C$  (e.g. if  $C$  operates at full capacity, it checks whether the received power is increased if it is not operational). The algorithm terminates if there is no such charger  $C$ .

For a given placement of a family  $\mathcal{C}$  of chargers and a family  $\mathcal{R}$  of receivers, define  $\delta(\mathcal{C}, \mathcal{R}) \stackrel{\text{def}}{=} \min\{|P(\mathcal{C}(\mathbf{x}), \mathcal{R}) - P(\mathcal{C}(\mathbf{x}^{(j,a)}), \mathcal{R})| : \mathbf{x} \in \{0, 1\}^m, a \in \{0, 1\}, j \in [m]\}$ . In particular,  $\delta(\mathcal{C}, \mathcal{R})$  is the minimal increment in the total received power that can be incurred by a single iteration of **IterativeMaxPower**. In addition, notice that every such iteration takes  $O(m^3)$  time. Finally, given that the chargers and receivers satisfy the placement constraints of Subsection 3.1, a crude upper bound for the maximum total power is  $nm^2 \gamma \beta^2 \frac{4\pi^2}{\lambda^2} = O(nm^2)$ . Therefore, we have the following:

---

#### Algorithm 1: IterativeMaxPower

---

```

Input :  $dist, \mathcal{R}, \mathcal{C}$ , communication_range
Output:  $\mathbf{x}$ 
1 begin
2    $\mathbf{x} \in \{0, 1\}^m$  is a random initial charger
      configuration;
3   while
4      $\exists C_j \in \mathcal{C} : P(\mathcal{C}(\mathbf{x}), \mathcal{R}) < P(\mathcal{C}(\mathbf{x}^{(j,a)}), \mathcal{R})$ ,  $a \in \{0, 1\}$ 
5     do
6       choose randomly a charger  $C_j \in \mathcal{C}$ ;
7        $\mathcal{R}_{C_j} = 0$  ;
8       foreach  $R \in \mathcal{R}$  do
9         if  $dist(C_j, R) \leq \text{communication\_range}$  then
10           $\mathcal{R}_{C_j} = \mathcal{R}_{C_j} \cup R$ 
11          //at this point  $C_j$  communicates with  $R$ 
12          //and receives  $\mathbf{E}(\mathcal{C}(\mathbf{x}), R)$ ;
13        end if
14      end foreach
15       $\mathbf{x}_{C_j} = \arg \max_{a \in \{0, 1\}} P(\mathcal{C}(\mathbf{x}^{(j,a)}), \mathcal{R}_{C_j})$ ;
16    end while
17    return  $\mathbf{x}$ ;
18 end

```

---

**THEOREM 2.** *Given a family  $\mathcal{C}$  of  $m$  chargers and a family  $\mathcal{R}$  of  $n$  receivers that satisfy the placement constraints of Subsection 3.1, Algorithm **IterativeMaxPower** finds an optimal solution of **MAX-POWER** in  $O\left(\frac{1}{\delta(\mathcal{C}, \mathcal{R})} nm^5\right)$ .*

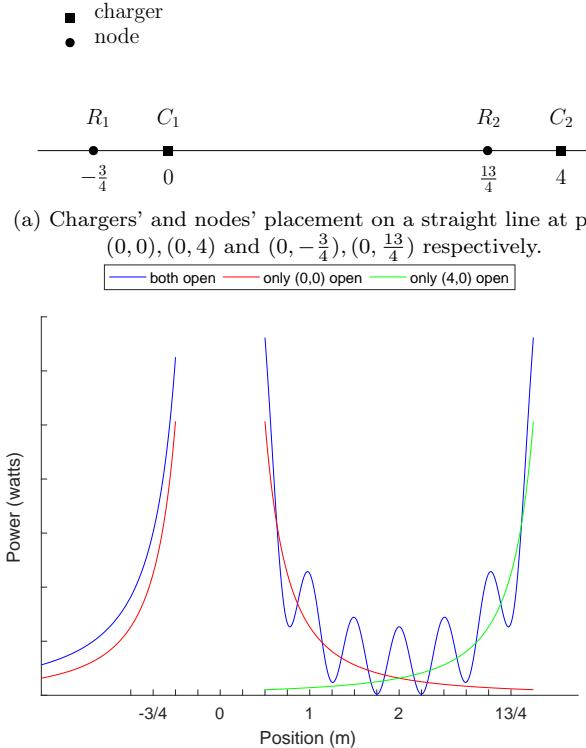
**Note:** In our experiments, we implemented **IterativeMaxPower** using different levels of knowledge of the wireless system. In particular, we define the *communication range* of a charger as the maximum radius of the disc area within which it can send and receive *messages* from nodes. Hence, a transmitter ignores any node that is outside its communication range. Whenever a charger  $C_j$  communicates with a node  $R$ , the latter sends to  $C_j$  the energy field vector  $\mathbf{E}(\mathcal{C}(\mathbf{x}), R)$ , where  $\mathbf{x}$  is the configuration at the time when the communication took place. This information is enough for the charger to compute  $P(\mathcal{C}(\mathbf{x}^{(j,a)}), R)$ , for each  $a \in \{0, 1\}$ , since  $P(\mathcal{C}(\mathbf{x}^{(j,a)}), R) = \gamma \|\mathbf{E}(\mathcal{C}(\mathbf{x}), R) + (a - \mathbf{x}_j) \mathbf{E}(C_j, R)\|^2$ . By using the above it is easy to compute  $P(\mathcal{C}(\mathbf{x}^{(j,a)}), \mathcal{R}_{C_j})$  for each  $a \in \{0, 1\}$ , where  $\mathcal{R}_{C_j} \subseteq \mathcal{R}$  includes the nodes in the communication range of the charger  $C_j$ . The pseudocode of **IterativeMaxPower** can be found in Algorithm 1.

## 6. MAXIMUM $K$ -MINIMUM GUARANTEE

In this section, we present our algorithmic solutions to **MAX- $k$ MIN-GUARANTEE**. This is more general than **MAX-POWER** and, even though we believe that it is computationally hard, we were unable to prove this formally. It is worth noting that the hardness of this problem does not lie in the computation of the minimum power among all  $k$ -set of receivers for a given configuration  $\mathbf{x}$ . It is not hard to see that this quantity is equal to the sum of powers of nodes having the  $k$  minimum powers.

On the other hand, exhaustive search algorithms do not work in the general case. In particular, Lemma 1 does not apply, as we show in the following example in which it is established that fractional operation power level of the transmitters can achieve better performance than  $x \in \{0, 1\}^m$ .

Assume again that there are two transmitters  $C_1$  and  $C_2$  placed at points  $(0, 0)$  and  $(4, 0)$  and two receivers  $R_1$  at  $(-\frac{3}{4}, 0)$  and  $R_2$  at  $(\frac{13}{4}, 0)$  respectively (see Fig. 2a). Also, assume that all constants in the above model are set to 1, i.e.  $\lambda = \beta = \gamma = 1$ . Fig 2b demonstrates the example on how the fractional operation level of  $C_1$  can increase the  $\min\{P(\{C_1, C_2\}, \{R_1\}), P(\{C_1, C_2\}, \{R_2\})\}$ . So, if  $x \in \{0, 1\}^2$  there are three operation cases (obviously when both chargers are closed is excluded). Firstly, if only  $C_1$  is on, then  $P(C_1, R_1) = (\frac{4}{3})^2 = 1.77$  and  $P(C_1, R_2) = (\frac{4}{13})^2 = 0.094$ . On the other hand if  $C_2$  is only on, the  $\min\{P(C_2, R_1), P(C_2, R_2)\} = (\frac{1}{4})^2 = 0.0625$ . In the final case, where  $x_{C_1} = x_{C_2} = 1$ , the power at the points is  $P(\{C_1, C_2\}, R_1) = (\frac{4}{3} + \frac{4}{13})^2 = 2.38$  and  $P(\{C_1, C_2\}, R_2) = (\frac{4}{3} - \frac{4}{13})^2 = 1.025$ . Clearly if the power from  $R_1$  is reduced by a small amount  $e$ , then  $P(\{C_1, C_2\}, R_2)$  achieves a better minimum and at the same time  $P(\{C_1, C_2\}, R_1)$  remains high.



(b) The power distribution in the straight line from the two chargers. Different curves represent different operation levels of the chargers.

Figure 2: Counter-example for fractional operation level of the chargers in MAX-kMIN-GUARANTEE problem.

In view of the above hardness indications, we consider a relaxation of MAX- $k$ MIN-GUARANTEE in which we are only interested in optimal configurations in  $\{0, 1\}^m$ , i.e. configurations in which each charger is either full operational, or does not operate.

**Optimal configuration in  $\{0, 1\}^m$ .** We consider the following exhaustive search solution, which we use as a measure of comparison for our heuristics. In particular, *Optimal Algorithm (OPT)* uses brute force to find an optimal solution. Due to its high time complexity  $O(2^m)$ , it is not practical

---

### Algorithm 2: Brute-force (OPT)

---

```

Input :  $dist, \mathcal{R}, \mathcal{C}, k$ 
Output:  $\mathbf{x}^{OPT}$ 
1 begin
2   |  $max\_power\_of\_k\_set = 0;$ 
3   | foreach  $\mathbf{x} \in \{0, 1\}^m$  do
4   |   | foreach  $R \in \mathcal{R}$  do
5   |   |   |  $\mathbf{p}(R) = P(\mathcal{C}(\mathbf{x}), R);$ 
6   |   | end foreach
7   |   |  $sort(\mathbf{p}, asc);$ 
8   |   | if  $(\sum_{i=1}^k \mathbf{p}(i) \geq max\_power\_of\_k\_set)$  then
9   |   |   |  $max\_power\_of\_k\_set = \sum_{i=1}^k \mathbf{p}(i);$ 
10  |   |   |  $\mathbf{x}^{OPT} = \mathbf{x};$ 
11  |   | end if
12  | end foreach
13  | return  $\mathbf{x}^{OPT};$ 
14 end

```

---

### Algorithm 3: Greedy (GRE)

---

```

Input :  $dist, \mathcal{R}, \mathcal{C}, k$ 
Output: Chargers configuration
1 begin
2   | choose an arbitrary  $\mathbf{x} \in \{0, 1\}^m$ ;
3   |  $y = a random chargers sequence;$ 
4   | for  $i = 1 \rightarrow length(y)$  do
5   |   | foreach  $R \in \mathcal{R}$  do
6   |   |   |  $\mathbf{p1}(R) = P(\mathcal{C}(\mathbf{x}^{(y(i),1)}), R);$ 
7   |   |   |  $\mathbf{p0}(R) = P(\mathcal{C}(\mathbf{x}^{(y(i),0)}), R);$ 
8   |   | end foreach
9   |   |  $sort(\mathbf{p1}, asc);$ 
10  |   |  $sort(\mathbf{p0}, asc);$ 
11  |   | if  $(\sum_{j=1}^k \mathbf{p0}(j) \leq \sum_{j=1}^k \mathbf{p1}(j))$  then
12  |   |   |  $\mathbf{x}_{y(i)} = 1;$ 
13  |   | else
14  |   |   |  $\mathbf{x}_{y(i)} = 0;$ 
15  |   | end if
16  | end for
17  | return  $\mathbf{x}$ 
18 end

```

---

when the problem size tends to grow. OPT can serve as a performance upper bound when benchmarking our algorithms.

Algorithm 2 crushingly searches (without taking anything into consideration) among all the possible configurations of the chargers the one that maximizes the cumulative power of the  $k$ -set of nodes with the least power received.

**Greedy Algorithm (GRE):** The algorithm initiates from a random configuration which through an iterative process improves. The algorithm's decision for the chargers' operation level is the one that contributes more to the cumulative received power of the  $k$ -set that consist of the nodes with the least power received. Although the heuristic disambiguation can sometimes perform badly, it may yield locally an optimal solution that approximates the global optimum in reasonable time. Clearly, time complexity depends on the number the algorithm checks to calibrate the chargers.

Algorithm 3 tries to find an optimal solution after a num-

---

**Algorithm 4:** Sampling (SAM)

---

**Input** :  $dist, \mathcal{R}, \mathcal{C}, k$   
**Output:**  $\mathbf{x}$

```

1 begin
2   choose randomly  $\sigma$   $k$ -sets of nodes  $\{k_1, k_2, \dots, k_\sigma\}$ ;
3   for  $i = 1 \rightarrow \sigma$  do
4     |  $\mathbf{x}^i = \text{IterativeMaxPower}(dist, k_i, \mathcal{C}, open)$ ;
5   end for
6    $perm = a \text{ random permutation of chargers}$ ;
7   for  $j = 1 \rightarrow m$  do
8     |  $gain\_0 = 0$ ;
9     |  $gain\_1 = 0$ ;
10    for  $i = 1 \rightarrow \sigma$  do
11      |  $\mathbf{p0}(i) = P(\mathcal{C}(\mathbf{x}^{i((perm(j),0)}), k_i)$ ;
12      |  $\mathbf{p1}(i) = P(\mathcal{C}(\mathbf{x}^{i((perm(j),1)}), k_i)$ ;
13      | if ( $\mathbf{p1}(i) \leq \mathbf{p0}(i)$ ) then
14        |   |  $gain\_0 = gain\_0 + \mathbf{p0}(i) - \mathbf{p1}(i)$ ;
15      | else
16        |   |  $gain\_1 = gain\_1 + \mathbf{p1}(i) - \mathbf{p0}(i)$ ;
17      | end if
18    end for
19    if ( $gain\_1 \leq gain\_0$ ) then
20      |   |  $\forall i \in [\sigma], \mathbf{x}_{perm(j)}^i = 0$ ;
21    else
22      |   |  $\forall i \in [\sigma], \mathbf{x}_{perm(j)}^i = 1$ ;
23    end if
24  end for
25  return any  $\mathbf{x}^i : i \in [\sigma]$ 
26 end

```

---

ber of iterations. In each step, it randomly chooses a charger  $y(i)$ , computes the power received from each node  $R$  and stores them in the vector  $\mathbf{p1}$  when  $y(i)$  is activated and at vector  $\mathbf{p0}$  when is deactivated. Subsequently, GRE measures the cumulative received power of the  $k$  nodes that receive the less power from  $\mathbf{p1}$  and  $\mathbf{p0}$  respectively. The operation level that is providing the larger amount of cumulative received power will be the algorithm's choice for the current step.

**Sampling Algorithm (SAM):** Instead of going through all possible solutions extensively as the Brute-force algorithm does, and in order to avoid the greedy approach and its disadvantages, we propose a sampling heuristic. It randomly samples  $\sigma$   $k$ -sets of nodes and aims to maximize the cumulative received power of all possible  $k$ -sets with the configuration that will come up from the sample. In this way, the algorithm overcomes the threat of a locally optimal solution but due to the random grouping, it doesn't take into account the nodes that form the  $k$ -set with the least received power.

Algorithm 4 consists of two phases. During the first phase it randomly chooses  $\sigma$   $k$ -sets of nodes and finds their optimal chargers configuration  $\mathbf{x}^i : i \in [1, \sigma]$  via Algorithm 1. The second phase is iterative. Within each iteration, it chooses a charger  $perm(j)$  from a random permutation ( $perm$ ) of the chargers set and measures the cumulative received power of each  $k$ -set that belongs to the sample when the charger is activated and when it is not. Then, it sums the gain when the charger is activated ( $gain\_1$ ) and compares

---

**Algorithm 5:** Fusion (FUS)

---

**Input** :  $dist, \mathcal{R}, \mathcal{C}, k$   
**Output:** Chargers configuration

```

1 begin
2   //find optimal configuration for each node;
3   foreach  $R \in \mathcal{R}$  do
4     |  $\mathbf{x}^R = \text{IterativeMaxPower}(dist, R, \mathcal{C}, open)$ ;
5   end foreach
6    $perm = a \text{ random permutation of chargers}$ ;
7   for  $j = 1 \rightarrow m$  do
8     foreach  $R \in \mathcal{R}$  do
9       |  $\mathbf{p0}(R) = P(\mathcal{C}(\mathbf{x}^{R((perm(j),0)}), R)$ ;
10      |  $\mathbf{p1}(R) = P(\mathcal{C}(\mathbf{x}^{R((perm(j),1)}), R)$ ;
11    end foreach
12    sort( $\mathbf{p1}$ , asc);
13    sort( $\mathbf{p0}$ , asc);
14    if ( $\sum_{i=1}^k \mathbf{p1}(i) \leq \sum_{i=1}^k \mathbf{p0}(i)$ ) then
15      |  $\forall R \in \mathcal{R}, \mathbf{x}_{perm(j)}^R = 0$ ;
16    else
17      |  $\forall R \in \mathcal{R}, \mathbf{x}_{perm(j)}^R = 1$ ;
18    end if
19  end for
20  return any  $\mathbf{x}^R : R \in \mathcal{R}$ 
21 end

```

---

it with the gain when it is closed ( $gain\_0$ ). The operation level with the larger sum of gain is decided as the final operation level of the charger. Thus, at the end of the process the  $\sigma$   $k$ -sets share the same chargers configuration. Finally, the configuration of the sampled  $k$ -sets is SAM's output.

**Fusion Algorithm (FUS):** Our last algorithm tries to combine the advantages of the previous ones and at the same time to restrict their weaknesses. FUS initiates by having the best chargers configuration for each node individually. Step by step it changes the operation level of one charger at the time. This way, the algorithm fuses the  $n$  different configurations to one with respect to the received power sum of the  $k$  nodes with the least power. FUS takes into consideration the received power for each node separately and aims to keep it as high as possible.

Algorithm 5 consists of two phases. During the first phase, it finds the optimal chargers configuration  $\mathbf{x}^R$  via algorithm 1 for each node  $R$  individually. During the second phase, it proceeds iteratively. Within each iteration, it chooses a charger  $perm(j)$  from a random permutation of the chargers set and measures the power of each node when it is activated and when its not. Those power values are stored in two vectors,  $\mathbf{p1}$  and  $\mathbf{p0}$  respectively. In the sequel, it sums the  $k$  nodes with the least power for each operation level and compares them. The operation level of the charger with the larger sum will be chosen. That way FUS changes the charger's operation level to the one that has the less negative impact on the  $k$ -set with the least received power that has risen. At the end of the algorithm, all the nodes share the same chargers configuration.

## 7. EVALUATION

We conducted simulations in order to evaluate our methods and reveal insights of the proposed design performance,

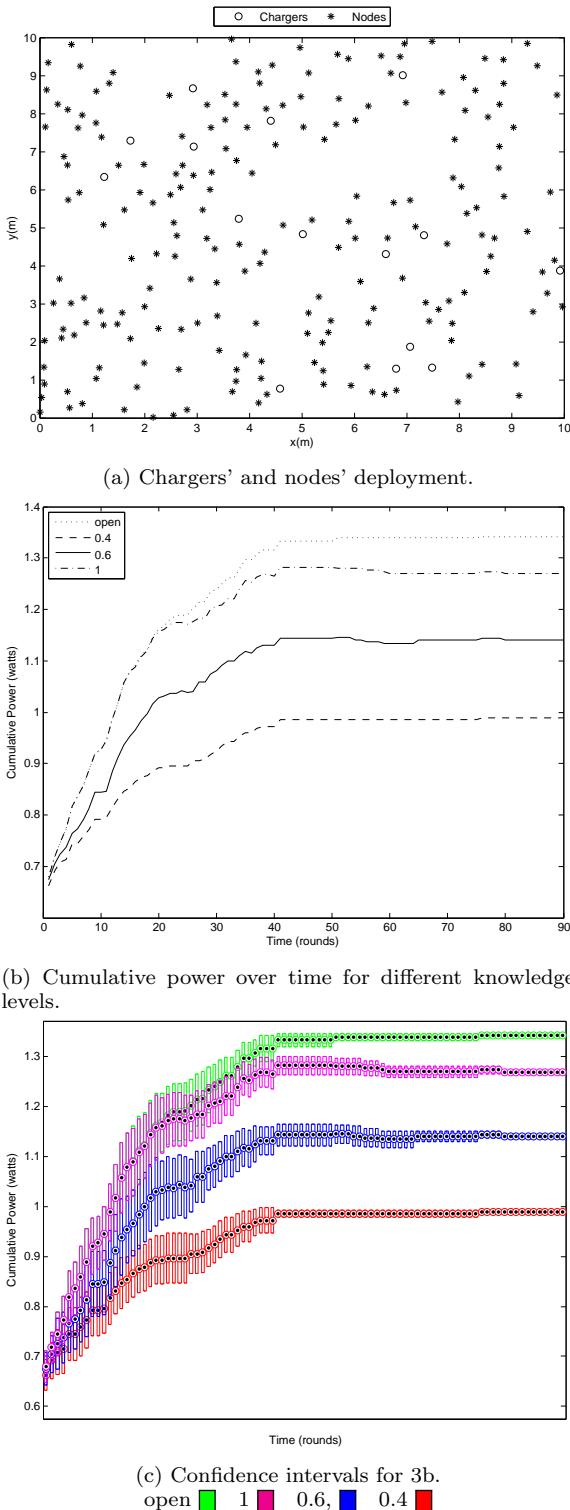


Figure 3: Deployment and cumulative power for Iterative-MaxPower.

using Matlab R2016a. The system that we consider for the performance evaluation consists of chargers and nodes randomly deployed in a square field of 10m x 10m. The number of chargers and nodes is set to 15 and 200 respectively and all the experiments run on the same system where the wavelength is a few centimeters. Fig 3a depicts an instance of the deployment. For statistical smoothness, we conducted each simulation 100 times. Even though the statistical analysis of the findings demonstrates very high concentration around the mean, in the following simulation results we also depict the confidence intervals. Actually, we provide the confidence intervals for 20 repetitions, in order to demonstrate an earlier convergence.

In this section, we provide our simulation results on three performance metrics: (a) cumulative power fuelled into the system, (b) communication overhead and (c) power balance (the variation of power among nodes).

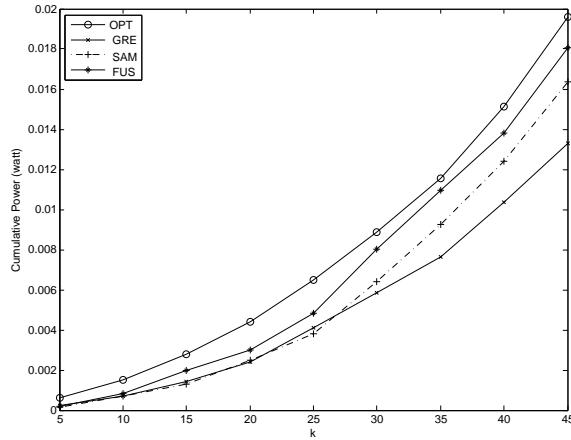
**Cumulative Power:** In general, as the communication range of the chargers grows, the distributed algorithm 1 achieves a near optimal solution. Fig 3b depicts the cumulative received power by the nodes under varying the communication range of the chargers over time (for 90 rounds). In each round, a charger is chosen randomly and decides on its operation level.

We can see that the cumulative power with the open communication range never decreases over time like the others do. This is because every charger has global knowledge of the power exchange in the system. Indeed, when a charger has limited communication range, then its choice serves the nodes in the communication range that covers, but for the rest of the nodes the result might be negative. On the other hand, at the first steps of the distributed algorithm, choices made from chargers with short communication range can benefit temporarily the cumulative power for the next steps, but in the end, the one with the global knowledge will perform better.

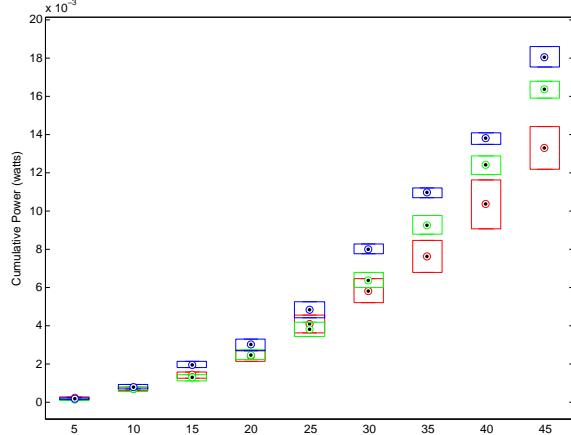
We also observe that when the communication range is 1m we achieve a near optimal solution. The reason behind this, is that the received power of the nodes far from the charger is reduced due to distance and they don't contribute that much to the cumulative received power. The confidence intervals which are presented in the Fig 3c show a high concentration around the mean. As the time passes algorithm 1 has higher concentration around the mean and without overlaps unlike at the first steps.

As mentioned before, it is impractical to run OPT because it uses brute force approach. For this reason, we use the same experimental setup with a small number of chargers as above for comparing GRE, SAM and FUS with OPT. Fig 3a shows the corresponding deployment.

We are also interested in the impact of the number  $k$ . Fig 4a depicts the cumulative power of the  $k$  nodes with the least power over various  $k$ . An interesting observation is that SAM doesn't perform much better than GRE for low  $k$ . On the other hand for  $k > 27$  we have a considerable improvement. The reason behind this is the sampling that SAM does. On the contrary, the rest of the algorithms compare the  $k$  least power values of all the system without any restriction. So for small  $k$  SAM by default has a disadvantage. For example, it is hard to sample in the same  $k$ -set for  $k = 10$  even the two of the nodes with the least received power among 200 nodes. Thus, as  $k$  increases, the performance of SAM improves. At the experiments  $\sigma = 30$  (the



(a) Cumulative power (of  $k$ -set with minimum power) for different values of  $k$ .

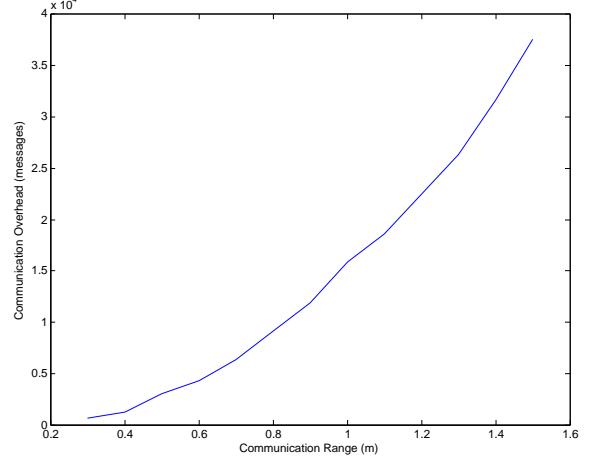


(b) Confidence intervals for 4a.  
GRE SAM FUS

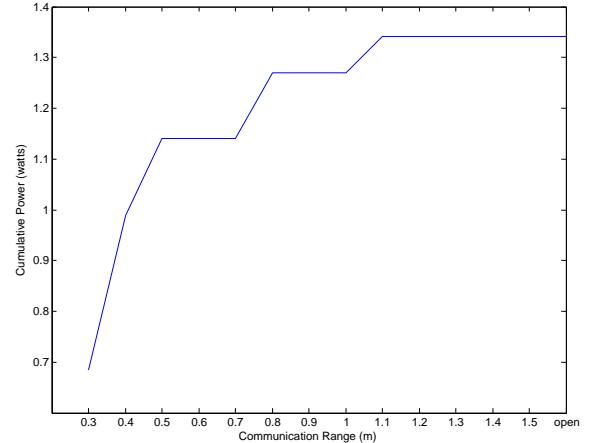
Figure 4: Performance of different algorithms.

size of the sample for SAM). Different values of  $\sigma$  didn't provide any significant improvement, but if it is too small the sample is not representative and poor. As expected, FUS outperforms the other two algorithms and it is the one that approximates OPT better. This is because FUS checks the power received for all the nodes and has more than one initial chargers' configuration unlike GRE. The confidence intervals are also presented in the Fig 4b. As we claimed, the algorithms present high concentration around the mean and there are no overlaps as the value of  $k$  grows.

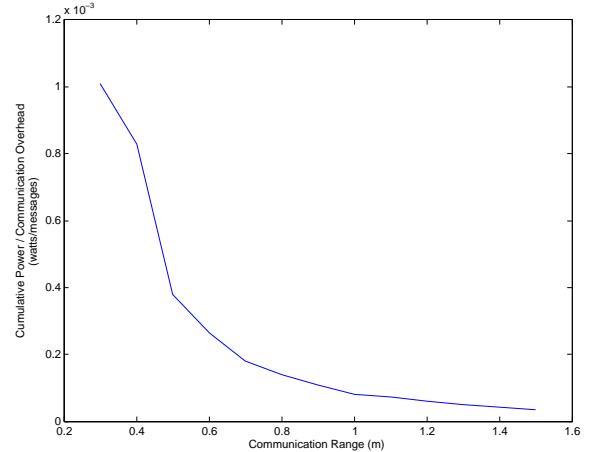
**Communication Overhead:** Fig 5a meters the number of messages for ranges from 0.3 to 1.5 that have been exchanged in the system during the running time of the distributed algorithm. When the communication range of the chargers increases, the communication overhead goes up. Moreover, the effect of different communication ranges (from 0.3 to 1.5 and *open*) on the cumulative received power of the system is depicted in Fig 5b. As we mentioned above a limited communication range ( $1.1m - 1.5m$ ) can achieve the performance of *open*. We observe that there is a trade-off between the communication overhead and the cumulative power that the nodes receive (Fig 5c). As the commun-



(a) Communication overhead over different communication ranges.



(b) Cumulative power over different communication ranges at the 90th round of Algorithm 1.



(c) Ratio of cumulative power to communication overhead over different communication ranges.

Figure 5: Impact of communication range on different metrics.

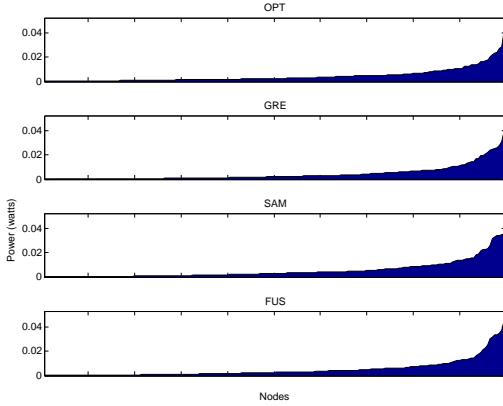


Figure 6: Power balance achieved by different algorithms.

cation range grows the ratio of cumulative received power to communication overhead decreases. It is evident that the contribution of the messages drops for communication ranges bigger than  $1m - 1.2m$ . After that point, the contribution of the extra messages, which are many, is very low.

**Power Balance:** We finally study the impact of our methods on the system in relation to power balance (the variation of power among the various nodes). The simulation results are shown in Fig 6 where we examine the performance of the algorithms in the general case. Obviously, there is room for improvement in this area, as our methods do not take power balance into consideration and we leave this as an open problem for future work.

## 8. CONCLUSION AND FUTURE WORK

In this paper, we studied the problem of finding the best configuration setup of the wireless power transmitters in a wireless system with respect to the maximization of the cumulative received power and we proposed heuristic algorithms to succeed in this goal. Finally, we evaluated the performance of the proposed algorithms through experimental simulation, and provided numerical results to validate their efficiency. A main contribution of our work lies on the fact that, for the first time power maximization algorithms are given under the vector model which realistically addresses the superposition of energy fields.

In future work, we opt to improve the power balance of the algorithms in exchange of a small amount of the cumulative received power. We will also explore solutions with good approximation ratios to maximize the cumulative received power from the point of view of nodes or chargers deployment.

## 9. REFERENCES

- [1] Powercast TX91501 Transmitter. <http://www.powercastco.com/products/powercaster-transmitters/>.
- [2] TI BQ501210 Wireless Power Transmitter Manager. <http://www.ti.com/product/BQ501210>.
- [3] C. M. Angelopoulos, S. Nikoletseas, and T. P. Raptis. Wireless energy transfer in sensor networks with adaptive, limited knowledge protocols. *Computer Networks*, 70:113 – 141, 2014.
- [4] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.
- [5] H. Dai, Y. Liu, G. Chen, X. Wu, and T. He. Scape: Safe charging with adjustable power. In *Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on*, pages 439–448, June 2014.
- [6] B. Griffin and C. Detweiler. Resonant wireless power transfer to ground sensors from a uav. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2660–2665, May 2012.
- [7] S. Guo, C. Wang, and Y. Yang. Mobile data gathering with wireless energy replenishment in rechargeable sensor networks. In *INFOCOM, 2013 Proceedings IEEE*, pages 1932–1940, April 2013.
- [8] S. He, J. Chen, F. Jiang, D. K. Y. Yau, G. Xing, and Y. Sun. Energy provisioning in wireless rechargeable sensor networks. *IEEE Transactions on Mobile Computing*, 12(10):1931–1942, Oct 2013.
- [9] Y. Li, L. Fu, M. Chen, K. Chi, and Y. H. Zhu. Rf-based charger placement for duty cycle guarantee in battery-free sensor networks. *IEEE Communications Letters*, 19(10):1802–1805, Oct 2015.
- [10] A. Madhja, S. Nikoletseas, and T. P. Raptis. Distributed wireless power transfer in sensor networks with multiple mobile chargers. *Computer Networks*, 80:89 – 108, 2015.
- [11] A. Mittleider, B. Griffin, and C. Detweiler. *Experimental Analysis of a UAV-Based Wireless Power Transfer Localization System*, pages 357–371. Springer International Publishing, Cham, 2016.
- [12] M. Y. Naderi, K. R. Chowdhury, and S. Basagni. Wireless sensor networks with rf energy harvesting: Energy models and analysis. In *2015 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1494–1499, March 2015.
- [13] M. Y. Naderi, K. R. Chowdhury, S. Basagni, W. Heinzelman, S. De, and S. Jana. Experimental study of concurrent data and wireless energy transfer for sensor networks. In *2014 IEEE Global Communications Conference*, pages 2543–2549, Dec 2014.
- [14] S. Nikoletseas, T. P. Raptis, and C. Raptopoulos. Low radiation efficient wireless energy transfer in wireless distributed systems. In *Distributed Computing Systems (ICDCS), 2015 IEEE 35th International Conference on*, pages 196–204, June 2015.
- [15] S. Nikoletseas, T. P. Raptis, and C. Raptopoulos. Interactive wireless charging for energy balance. In *Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on*, June 2016.
- [16] L. Xie, Y. Shi, Y. T. Hou, and H. D. Sherali. Making sensor networks immortal: An energy-renewal approach with wireless power transfer. *IEEE/ACM Transactions on Networking*, 20(6):1748–1761, Dec 2012.
- [17] S. Zhang, Z. Qian, F. Kong, J. Wu, and S. Lu. P3: Joint optimization of charger placement and power allocation for wireless power transfer. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 2344–2352, April 2015.

# Are We Birds of the Same Feather?

**Neeharika Kompala**  
 Department of Information  
 Technology  
 Chaitanya Bharathi Institute of  
 Technology, Hyderabad  
 neeharika.369@gmail.com

**S.R.S. Iyengar**  
 Department of Computer  
 Science  
 Indian Institute of Technology  
 Ropar  
 sudarshan@iitrpr.ac.in

**Yayati Gupta**  
 Department of Computer  
 Science  
 Indian Institute of Technology,  
 Ropar  
 yayati.gupta@iitrpr.ac.in

## ABSTRACT

The paper addresses the community membership problem of predicting whether two nodes belong to the same community or not, using a hill climbing based breadth first traversal algorithm and a probabilistic depth first traversal algorithm. The algorithms have lesser time complexity as compared to the standard community detection algorithms. The proposed algorithms achieve a high efficiency in terms of the fraction of node pairs correctly identified, which is greater than 70% in most of the network instances. Moreover, a very little fraction of the graph is traversed which is local to the nodes in consideration.

## Keywords

Community detection, breadth first traversal, depth first traversal

## 1. INTRODUCTION

"Birds of a feather flock together" [23], "Like Attracts Like", yet in other terms Homophily, is a well known phenomenon in the field of complex networks. This phenomenon gives rise to a meso-scale structure, specifically community structure. Informally speaking, communities are the small denser subgraphs in a network, loosely connected to each other. One can visualise the community structure by superimposing the given network over an erdos-renyi graph having the same number of nodes and edges. While the edges in a random graph are uniformly distributed, the edges in the given network form dense clusters at some areas while leaving other areas sparse. The denser areas correspond to one community, with the edges here called intra-community edges. The edges in the sparse areas are called inter-community edges. Communities are seen almost everywhere- educational institutes, companies [33], world wide web [15], protein-protein interaction networks [5], metabolic networks [34], political systems, purchase networks, network of cells in human body, citation networks [6], collaboration networks etc. with social networks [11] remaining their paradigmatic examples.

Community detection, i.e. unveiling the community structure in a network, stands as one of the most celebrated problems in the area of complex networks. It provides us important information about

the structure and functionality of the network. If an online retailer knows the community structure in a network, she might better predict the preferences of the people. If a deadly disease is spreading in a network, curbing it at the community level might curb it at the global level also. According to Weng et al. [36], it is possible to predict the virality of a meme by looking at its distribution across different communities. A viral meme tends to infect multiple communities, while a non viral meme remains trapped inside its own community because of social reinforcement and homophily. Hence, community categorisation of nodes holds a large number of applications in diverse fields ranging from economics to biology.

Imagine a country consisting of 1000 political parties. We meet two politicians on the way. We are interested in finding whether these two politicians belong to the same party or not. One way to solve this problem is to obtain the entire network of politicians in the country, apply community detection algorithm to it, then see whether they belong to the same community or not. But there tend to be two problems with this approach:

- The entire political network might not be available. Rather, it is possible to discover a smaller part of the network by asking these politicians about their local network. The global picture is inaccessible in most of the real world scenarios.
- Assume, there are 1000 communities in this political network. Revealing these 1000 communities and then declaring the membership is a time consuming approach. The main problem is just to know whether the two politicians are in the same party or not, which might be solved with a much lesser complexity.

Similarly, in a friendship network among the students of a school, it might be sufficient to know whether two students belong to the same class or not. In protein-protein interaction networks, to see whether these two proteins serve the same function or not also does not require unravelling the entire community structure. Same holds for finding out whether two WWW pages belong to the same topic or not. In the viral meme example, if there are 3 infected nodes, our aim is just to know whether they belong to the same community or not. For this, one need not run the time complex community detection algorithm. Moreover, community detection algorithms demand the presence of the entire network at hand. Generally, in many real world applications, a smaller question needs to be addressed. In this paper, we propose two algorithms, which without exploring the complete network, and with much lesser time complexity answer the membership question.

The major contributions of the paper are :

ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of a national government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only.

ICDCN '17, January 04-07, 2017, Hyderabad, India

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007753>

- Predicting whether the given two nodes belong to the same community or not without using the time consuming community detection algorithms.
- The prediction is based on the local information of the nodes. Hence, the algorithm is successful even when the entire network is not available to us.

The problem statement, now, can be formally stated as follows: Given an undirected graph  $G(V, E)$  with non-overlapping communities and two nodes  $i$  and  $j$  such that  $i, j \in V(G)$ , find  $\delta(i, j)$ , where

$$\delta(i, j) = \begin{cases} 1 & \text{if } i \text{ and } j \text{ belong to the same community} \\ 0 & \text{otherwise} \end{cases}$$

The rest of the paper is organised as follows. Section 2 lays a light on the research work already done towards this direction. Though, to the best of our knowledge, no one has tried to address the exact problem as ours, but there are works whose techniques come closer to ours. They are elaborated in this section. Section 3 builds a preliminary approach towards attacking the problem. It also describes the datasets and some experiments which helped us converge towards the idea. Section 4 formally defines the algorithms we are using. Section 5 mentions the results of the proposed algorithms both in terms of efficiency and accuracy. Finally, the paper is concluded in section 6 along with the future directions that one can pursue.

## 2. RELATED WORK

The literature of community detection algorithms in the field of complex networks is a very rich one. Community structure is observed in both- natural as well as artificial networks.

Most of the community detection algorithms be they divisive, agglomerative, spectral, genetic or greedy require the knowledge of complete network structure. Graph partitioning algorithms need the entire network [2, 14, 27, 30]. Hierarchical community detection algorithms can be categorised as agglomerative or divisive. The most popular divisive algorithm is the one proposed by Girvan and Newman [11]. The edge clustering coefficient method proposed by Radichhi et al. [28] and information centrality method by Latora and Marchiori [18] are similar to the Girvan and Newman's method and hence require the global information of the network. The greedy hierarchical clustering method [25], though agglomerative in nature, requires the knowledge of the complete graph. The loop coefficient method proposed by Vragovic and Louis [35] works by building communities around the identified core nodes. Many other algorithms like external optimisation [3, 9], spectral optimisation [29, 31, 37] as well as genetic algorithms [24, 32] are based on maximisation of the modularity function, which is global in nature. Lin et al. [22] proposed a community detection algorithm for the incomplete information networks, where some of the edges are missing. While most of the scientists used only global information for detecting community structure, many have tried to amalgamate global and local information [8] for this purpose. Research has also been done on detecting the community outliers [10].

Our method resembles the well known seed set expansion techniques used for the identification of communities in a local fashion. Most of these algorithms build around a set of seed nodes to discover the entire community. Some of the approaches in this direction are truncated random walk [1], unnormalised page rank [16]

based on the modified google page ranking for personalised web search, greedy approaches which try to maximise the minimal degree [7] or the number of links the prospective neighbours have with the current members of the community [22]. Most of these approaches start from a set of seed nodes. This set plays an important role in defining the efficiency of these algorithms. Hence, seed set optimisation is another problem in this context. Unlike these algorithms, we are given only two seed nodes which might belong to different communities also. Our aim differs from the conventional algorithms in the sense that we do not wish to identify the communities of these nodes accurately, rather just declare whether they belong to the same community or not.

There has been a lot of work towards community detection. However, to the best of our knowledge, no one has tried to address the smaller problem of membership explicitly, which requires lesser resources, both in terms of time as well as information.

## 3. THE IDEA

Consider the city of Hyderabad. There are many sectors in this city. Let there be two students, Alia and Bijesh, interested in finding whether they belong to the same sector or not. Alia starts exploring local nearby places. Bijesh does the same. An already visited place is not visited again. When they are done exploring a sufficient fraction of their sectors, they stop. Now, the number of common places (explored by both of them) gives us an idea of whether they belong to the same sector or not. The idea has been elaborated in the following points.

- Start exploring the network locally from the nodes  $i$  and  $j$ .
- Stop the exploration upon reaching a termination condition.
- The explorations from  $i$  and  $j$  result in the formation of trees which we name  $T_i(V_i, E_i)$  and  $T_j(V_j, E_j)$  respectively.
- Compute  $N = |V(T_i) \cap V(T_j)|$ . Greater the  $N$ , higher the probability that both nodes belong to the same community and vice versa.

A similar approach can be taken to solve our problem. Starting exploration from the two seed nodes, we discover some potential nodes in their communities. An intersection of these two sets of nodes gives us our solution. An ideal solution is the one yielding accurate results while exploring as less fraction of the graph as possible.

Two questions that need to be answered in this context are

1. What should be the ideal exploration technique to discover new nodes?
2. What should be the stopping criterion for the explorers?

Before describing the exploration and stopping criteria, we describe the datasets used in the paper.

### 3.1 Datasets

We have used two benchmark networks and seven real world networks for the experimentation purposes. The benchmark networks have been derived from the benchmark [17] proposed by Fortunato et al. This benchmark generates networks having scale free degree distribution and community structure, with a number of parameters which are user defined. The parameters are minimum degree  $k_{min}$ ,

maximum degree  $k_{max}$ , power law exponent  $\gamma$ , minimal and maximal community sizes represented by  $s_{min}$  and  $s_{max}$  respectively and the mixing parameter  $\mu$ .

The real world networks taken for study [11, 19, 26, 38] are implicitly equipped with the properties of power law degree distribution and community structure.

The modularity for all the networks is calculated using the standard formula

$$Q = \frac{1}{2m} \sum_{ij} ((A_{ij} - P_{ij})\delta(C_i, C_j))$$

where  $A$  is the adjacency matrix of the given network,  $m$  is the edge count,  $P_{ij}$  is the expected number of edges between vertices  $i$  and  $j$  in the null model. A null model is the one that have the same number of edges as well as the same degree distribution as the original network, but the edges are laid in a different style than the random model. The  $\delta$  function is defined as :

$$\delta(i, j) = \begin{cases} 1 & \text{if } i \text{ and } j \text{ belong to the same community} \\ 0 & \text{otherwise} \end{cases}$$

The datasets used for experiments alongwith their properties are described in Table 1 and Table 2. We have considered a broad range of datasets. The Dolphin network represents the frequent associations between the dolphins which were living as a part of a community in New Zealand. The Karate Club network was obtained from a university in US in the year 1970. The network is between the members of a club based on their friendships. Next is the network of American Football games. Cooperfield network is the network of words formed from the novel - "David Copperfield" written by Charles Dickens. The words considered here are common adjectives and nouns and their adjacency gives rise to a complex network. Facebook data is the anonymised Facebook friendship network. Astro-ph is a coauthorship network based upon whether two authors have published a paper together. The data is collected from E-Print Archive from the year 1995 to 1999. Cond-mat is a similar collaboration network from the Condensed Matter E-Print archive. While the networks like Dolphin, Karate and Football are very small in size consisting hardly of 100 nodes, Facebook network has around 4000 nodes and collaboration networks have millions of nodes.

There are two benchmark networks we have used for our experiments having 2000 and 4000 nodes respectively. Their other parameters are given in Table 2.

Name	Number of Nodes	Number of Edges	Modularity
Dolphin	62	158	0.5165438231
Karate club	34	77	0.4174397032
Football	115	612	0.6054869388
Copperfield	112	424	0.2817061232
Facebook	4039	88233	0.8349721974
astro-ph	16706	121251	0.7259871359
cond-mat	16726	47594	0.84676577

Table 1: Real World Networks used for experiments

Parameter	Benchmark Network 1	Benchmark Network 2
Number of Nodes	2000	4000
Number of Edges	14044	50360
Modularity	0.580457	0.6952
Average Degree	14	7
Maximum Degree	500	900
Degree exponent	2	2
Community exponent	2	1
Mixing Parameter	0.2	0.2

Table 2: Benchmark Networks used for experiments

### 3.2 Criteria for Exploration and Stopping

The nodes having high betweenness act as gatekeepers for the information flow among communities [4]. Betweenness of a node is defined as the fraction of shortest paths running through this node. Assume  $\sigma_{u,v}$  denotes the number of shortest paths between the nodes  $u$  and  $v$ . Then the betweenness centrality of a node  $i$  can be defined as

$$\text{Betweenness}(i) = \frac{\sigma_{u,v}(i)}{\sigma_{u,v}}$$

where  $\sigma_{u,v}(i)$  denotes the number of shortest paths between  $u$  and  $v$  which passes through  $i$ .

Betweenness centrality can be normalised by dividing the above term with the number of node pairs excluding  $i$  i.e. dividing by  $\frac{(n-1) \times (n-2)}{2}$ . This sets the value of centrality in the range from 0 to 1.

Greater the number of shortest paths running through a node, more important the node is in controlling the flow across the given network. Removal of these nodes has a great impact on the connectivity of a network. Such nodes are mostly responsible for connecting the otherwise disconnected components in a network [13]. Thus, while taking a walk in a community, such nodes can be used as indicators of approaching the boundary of the community. This has been shown in Figure 1. The first network in the figure is an artificial network having an ideal community structure. We generated 4 complete networks with one having 6 nodes and others having 5 nodes each and then associated them by linking some random edges. The second network shown in the figure is the well known Zachary Karate club network which has 4 communities.

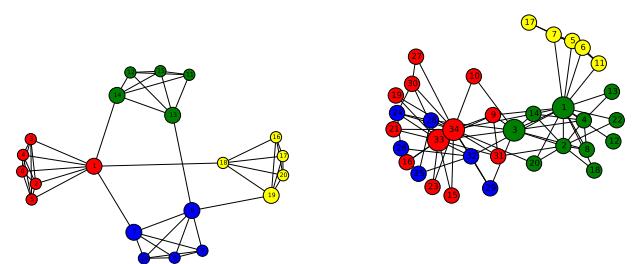


Figure 1: Location of high betweenness nodes in a complex network. Left: benchmark Network. Right: Zachary Karate Club Network

In this figure, different colours of nodes represent different communities. So, both the networks shown have 4 communities each. The nodes having high betweenness centralities are rendered bigger in size. Hence, it can be seen that the nodes having high betweenness centralities are the ones connecting communities.

But, betweenness is a global measure as one needs to see all the shortest paths present between every two pairs of nodes. Moreover, calculation of betweenness centrality for all the nodes takes  $O(n^3)$  time, where  $n$  is the number of nodes in the network. Below, we show the correlation of betweenness with the two most widely used local attributes of a node- degree and clustering coefficient.

### 3.2.1 Correlation with Degree Centrality

Let  $A_{ij}$  be the adjacency matrix of the graph  $G(V, E)$ , where

$$A_{ij} = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are connected by an edge in } E \\ 0 & \text{otherwise} \end{cases}$$

The degree of a node  $i = \sum_j A_{ij}$ . Degree centrality of a node is its normalised degree expressed as

$$\text{Deg}_i = \sum_j \frac{A_{ij}}{n-1}$$

In Figure 2, we show the highest degree nodes present in the artificial and the Zachary Karate network, as we have done previously. It can be seen that for the Karate club network, the topmost degree nodes are the same as the highest betweenness nodes which connect two communities with each other. In the artificial network, the top two highest degree nodes coincide with the top two highest betweenness nodes, but there is no particular relation for the lesser degree nodes. For the remaining nodes, degrees are approximately the same. The reason for this observation is the uniform degree distribution of the considered network, which is generally not the case in real world scenarios.

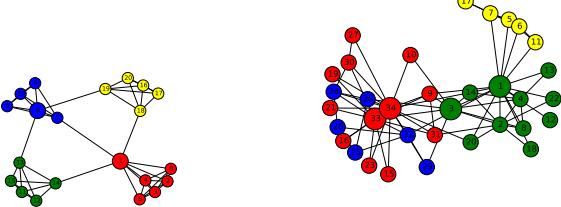


Figure 2: Location of high degree nodes in a complex network. Left: benchmark Network. Right: Zachary Karate Club Network

The correlations of betweenness with degree for one benchmark and one real network have been shown in Figure 3. The correlation for nine more networks can be seen in Figure 6. It can be seen that the nodes having high betweenness values also possess a high degree centrality though there is no peculiar relationship between the nodes having low betweenness and degree. But, for us, the nodes of interest are only the ones having high values of betweenness and degree.

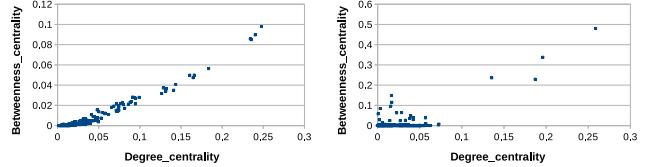


Figure 3: Correlation between degree and betweenness. Left: benchmark Network (SN1). Right: Facebook Network

In benchmark networks, all the nodes show positive correlation between clustering and betweenness. However, in real world networks, such a correlation is observed only for a few nodes which have very very high degree as well as betweenness. This observation is attributed to the lesser difference in the number of intra-community and inter-community links in benchmark networks as compared to the real world networks.

### 3.2.2 Correlation with Clustering Coefficient

The clustering coefficient of a node denotes the degree of friendship between its neighbours. A person who is tightly bounded in his own community tends to have a higher clustering coefficient. On the other hand, people who act as mediators between many communities tend to have a low clustering coefficient.

Let  $N(i)$  denote the set of neighbours of node  $i$ , i.e.  $N(i) = \{j : A_{ij} = 1\}$ . Then, Clustering coefficient for  $i$  can be expressed as follows

$$CC(i) = \frac{\sum_{j \in N(i)} \sum_{k \in N(i) \setminus \{j\}} A_{jk}}{\binom{|N(i)|}{2}}$$

The numerator in the above expression denotes the number of existing relationships between the friends of  $i$  and denominator depicts the total number of friendships actually possible.

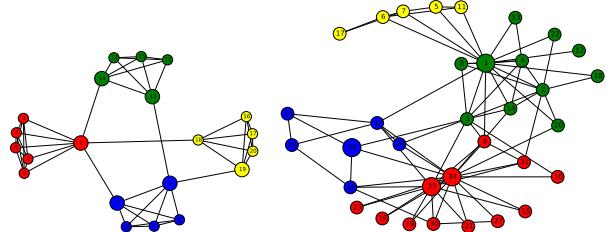


Figure 4: Location of high clustering nodes in a complex network. Left: benchmark Network. Right: Zachary Karate Club Network

The **correlation of betweenness with clustering** has been shown in Figure 4. In benchmark networks, all the nodes show negative

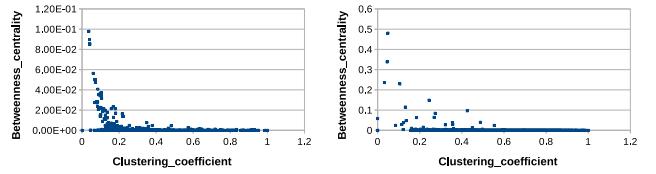


Figure 5: Correlation between clustering and betweenness. Left: Benchmark Network (SN1). Right: Facebook Network

correlation between clustering and betweenness. However, in real

world networks, such a correlation is observed only for a few nodes which have very low clustering but a high betweenness. This observation is attributed to the lesser difference in the number of intra-community and inter-community links in benchmark networks as compared to the real world networks. These correlations for the real world networks are also obtained by Goh et al. [12].

The correlations between clustering coefficient and betweenness centrality for some more networks have been shown in Figure 6.

### 3.2.3 Exploration Criterion

The exploration from the seed nodes is done according to the variants of Breadth First Traversal and Depth First Traversal, which are mentioned below:

- **Selective Breadth First Traversal(SBFT):** This approach is a hill climbing based breadth first traversal. There are multiple explorers working in different directions. Each child node of the root acts as an explorer. Hence, the number of explorers is equal to the degree of the seed node. We try moving inwards the community. To do so, the next node chosen is the neighbour having the minimum degree or the maximum clustering, based on the above observations. One can see this algorithm as a greedy hill climbing algorithm with multiple branches running independently except for the fact that an already visited node is not visited again. Once a branch reaches its local maxima (in case of clustering) or local minima (in case of degree), it stops growing.
- **Probabilistic Depth First Traversal(PDFT):** This approach adopts a depth first approach towards the same goal. But, since the local maxima or minima can be reached very quickly here, we do not want to completely ignore the semi-optimal neighbours. Hence, the probability of choosing a neighbour is taken to be inversely proportional to its degree.

### 3.2.4 Stopping Criterion

The stopping criterion for SBFT is when all the branches stop growing i.e. when all the branches reach their local maxima/minima. The stopping criterion for PDFT is when 100 nodes in the community are explored. We choose the threshold of 100, because it is established that on an average, the communities in the real world comprise of 100 nodes [20].

## 4. THE PROPOSED ALGORITHMS

The base algorithm, i.e. the membership algorithm takes as input two nodes  $i$  and  $j$ . It explores the graph starting from both the nodes, according to the rules of either 1) Selective Breadth First Traversal (SBFT) or 2) Probabilistic Depth First Traversal (PDFT) algorithm. These explorations lead to the formation of two trees, whose intersection is a predictor of  $\delta(i, j)$ . We look at the sets of vertices of both the trees. If the number of common vertices exceed the number of vertices which are the part of exactly one set, we declare the answer to be one, else zero. Algorithm 1 explains this. In all the algorithms,  $N(i)$  denotes the set of neighbours of node  $i$ .

### 4.1 Community Traversal Algorithms- SBFT and PDFT

SBFT algorithm proceeds like breadth first search algorithm, but only selected nodes are explored. The selected nodes are the ones having less degree (in the degree based approach) or a high clustering coefficient (in the clustering based approach) than their parents.

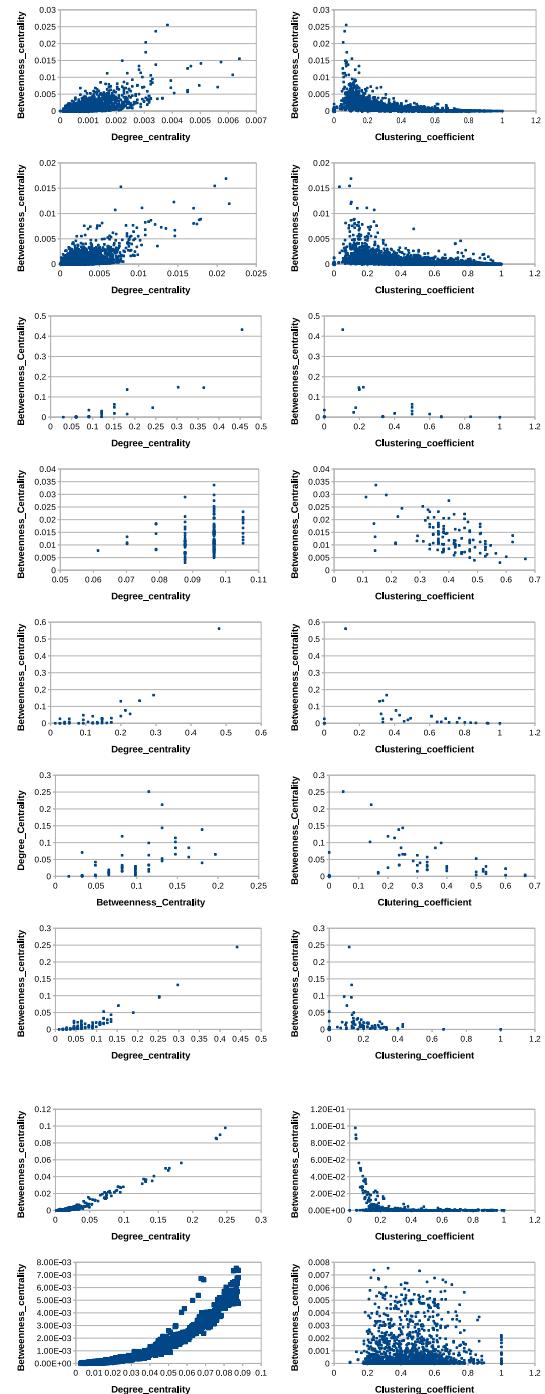


Figure 6: Correlation of betweenness centrality with degree centrality and clustering coefficient. Top to bottom: 1) Condmat Graph 2) Astrophysics Network 3) Karate Club Network 4) Football Network 5) Les Misérables Network 6) Dolphin Network 7) Copperfield Network 8) Benchmark Network 1 9) Benchmark Network 2

---

**Algorithm 1** The Membership Algorithm

---

```
1: procedure MEMBERSHIP
2:   Input:- Graph  $G(V,E)$ , Nodes  $i$  and  $j$ , parameter  $chc$ 
3:   Output:  $\delta(i,j) \in \{0,1\}$ 
4:   if  $chc=1$  then
5:      $T_1 \leftarrow SBFT(i)$  and  $T_2 \leftarrow SBFT(j)$ 
6:   end if
7:   if  $chc=2$  then
8:      $T_1 \leftarrow PDFT(i)$  and  $T_2 \leftarrow PDFT(j)$ 
9:   end if
10:  if  $|V(T_1) \cap V(T_2)| \geq |(V(T_1) \cup V(T_2)) - (V(T_1) \cap V(T_2))|$ 
then
11:    Return 1
12:  else
13:    Return 0
14:  end if
15: end procedure
```

---

SBFT ensures exploring at least one level from the seed nodes. Algorithm 2 describes this.

---

**Algorithm 2** Selective Breadth First Traversal Algorithm

---

```
1: procedure SBFT
2:   Input:- Graph  $G(V,E)$ , Root node  $root$ 
3:   Output: The tree  $T$  corresponding to  $root$ 
4:   Initially no node is visited. So, set node attribute
    $visited(u) \leftarrow 0 \quad \forall u \in V(G)$ 
5:    $visited(root) \leftarrow 1$ 
6:   Create a queue,  $Q \leftarrow \{\emptyset\}$ . Initially  $Q$  is empty.
7:    $Q.enqueue(root)$ 
8:    $t = (V(T), E(T))$ . Initially,  $V(T) \leftarrow \{\emptyset\}$  and  $E(T) \leftarrow \{\emptyset\}$ .
9:    $V(T).append(root)$ 
10:   $flag \leftarrow 0$ 
11:  while  $Q$  is not empty do
12:     $u \leftarrow Q.dequeue()$ .
13:    for  $v \in N(u)$  do
14:      if  $visited(v) = 0$  then
15:         $E(T) \leftarrow E(T) \cup (u, v)$ 
16:         $visited(v) \leftarrow 1$ 
17:        if  $flag = 0$  then
18:           $Q.enqueue(v)$ 
19:           $flag \leftarrow 1$ 
20:        else
21:          if  $degree(v) \leq degree(u)$  (in degree based
             approach) or  $clustering(v) \geq$ 
22:               $clustering(u)$  (in clustering based ap-
             proach) then
23:                 $Q.enqueue(v)$ 
24:              end if
25:            end if
26:          end if
27:        end for
28:      end while
29:      return  $T$ 
30: end procedure
```

---

PDFT algorithm is analogous to the Depth-First-Search algorithm, but the probability of picking a neighbour is inversely proportional to its degree. Moreover, the algorithm terminates on exploring at most 100 nodes [21]. The number 100 is chosen as the limit since most of the communities in the real world have a max-

imum size of 100 from the root. This is explained in algorithm 3.

## 5. RESULTS AND EXPERIMENTS

We measure the accuracy and efficiency of our algorithms as follows.

- **Accuracy:** The number of node pairs identified correctly by our algorithm determines its accuracy. Both, the number of true positives and true negatives contribute to the accuracy of the algorithm. Since, the number of node pairs possible are extremely large in some of the graphs, we randomly pick a sample of the pairs and check the accuracy of our algorithm over them.
- **Efficiency:** The overall purpose of the algorithm is to solve the proposed problem without exploring much of the network. That is determined by efficiency, which quantifies the fraction of network explored to solve the problem.

### 5.1 Accuracy

Table 3 shows the accuracy of the results of our algorithm on 5 networks. It can be seen that the algorithms scale well since the efficiency of the algorithm does not decrease on taking bigger networks, both in terms of degree and edges.

Name	Number of pairs sampled	Positives	Negatives	Accuracy
Dolphin	1824	1183	641	0.645
Karate	396	303	93	0.7651
Football	6439	5592	847	0.87
Copperfield	5808	2956	2852	0.5089
Facebook	100000	87006	12994	0.87

Table 3: Accuracy

It can be seen that the results for accuracy are closely related with the modularity of the networks. The copperfield network having the least modularity value (0.3 approximately) has the least accuracy. Dolphin network and the Karate club networks also have relatively lesser modularity and hence have intermediate accuracy results. Finally, the Facebook and Football networks have a very high modularity value and the accuracy in their case peaks to as high as 87 %. This denotes that the algorithm is working very efficiently. Provided one uses the correct method to identify communities on the network, our algorithms give 80% accurate results.

### 5.2 Efficiency

In PDFT algorithm, the fraction of graph traversed is mostly equal to the number of nodes in a community, i.e. maximum 200, if the nodes belong to different communities. In the following, we discuss the fraction of graph traversed in case of SBFT algorithm. Figure 7 shows the fraction of graph traversed for the Facebook network in the case of SBFT.

It can be seen that the distribution here follows a power law. Most of the pairs explore quite less fraction of the network, while only a few pairs explore a significant fraction of the network. Table 4 shows the minimum, maximum, average and standard deviation of the number of nodes traversed in the case of different networks, in the case of degree based SBFT.

Table 5 shows the same for clustering coefficient based SBFT.

---

**Algorithm 3** Probabilistic Depth First Traversal Algorithm

---

```

1: procedure PDFT
2:   Input:- Graph  $G(V,E)$ , Root node  $root$ 
3:   Output: The tree  $T$  corresponding to  $root$ 
4:   Initially no node is visited. So, set node attribute
    $visited(u) \leftarrow 0 \quad \forall u \in V(G)$ 
5:    $visited(root) \leftarrow 1$ 
6:    $numsteps \leftarrow 0$ 
7:   Create a stack,  $S \leftarrow \{\phi\}$ . Initially  $S$  is empty.
8:    $S.push(root)$ 
9:    $u \leftarrow root$ 
10:   $T = (V(T), E(T))$ . Initially,  $V(T) \leftarrow \{\phi\}$  and  $E(T) \leftarrow \{\phi\}$ .
11:   $V(T).append(root)$ 
12:  while  $numsteps < 100$  do
13:    if  $visited(w) = 1 \quad \forall w \in N(u)$  then
14:       $temp = S.pop()$ 
15:      while  $visited(x) = 1 \quad \forall x \in N(temp)$  do
16:         $temp = S.pop()$ 
17:      end while
18:    end if
19:     $neighbour\_set = \{w | w \in N(u)\}$ 
20:    Initialise  $inv\_deg\_list = []$ 
21:     $list = []$ ,  $cum\_list = []$  and  $sum\_inv\_deg = 0$ 
22:    for  $v \in neighbour\_set$  do
23:       $inv\_deg\_list.append(\frac{1}{deg(v)})$ 
24:       $sum\_inv\_deg = sum\_inv\_deg + \frac{1}{deg(v)}$ 
25:       $cum\_list.append(0)$ 
26:    end for
27:     $cum\_list.append(0)$ 
28:    for  $(i = 0$  to  $i = length(inv\_deg\_list) - 1)$  do
29:       $inv\_deg\_list[i] \leftarrow \frac{inv\_deg\_list[i]}{sum\_inv\_deg}$ 
30:    end for
31:    for  $(i = 0$  to  $i = length(inv\_deg\_list) - 1)$  do
32:       $cum\_list[i] \leftarrow cum\_list[i] + inv\_deg\_list[i]$ 
33:    end for
34:    while  $(1)$  do
35:      Pick  $r \in [0, 1]$ , where  $r \in R$ 
36:      for  $(i = 0$  to  $i = length(cum\_list) - 1)$  do
37:        if  $cum\_list[i] < r$  then
38:          continue
39:        else
40:          break
41:        end if
42:      end for
43:       $new\_u \leftarrow neighbour\_list[i]$ 
44:      if  $visited(new\_u) = 0$  then
45:         $visited(new\_u) = 1$ 
46:         $numsteps = numsteps + 1$ 
47:         $S.push(new\_u)$ 
48:         $E(T) \leftarrow E(T) \cup (u, new\_u)$ 
49:         $u \leftarrow new\_u$ 
50:        break
51:      else
52:        continue
53:      end if
54:    end while
55:  end while
56:  return  $T$ 
57: end procedure

```

---

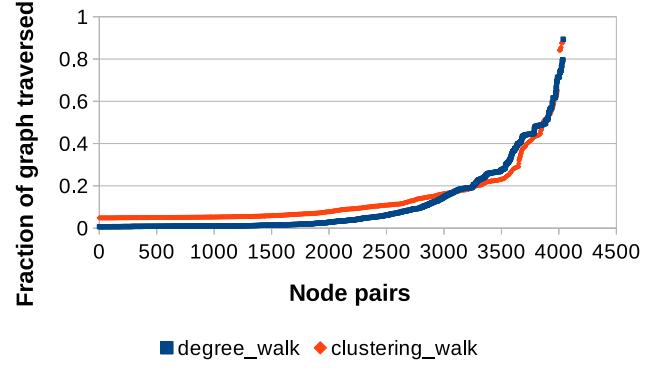


Figure 7: Fraction of graph traversed in Facebook Network

Name	Min	Max	Avg	StDev
Dolphin	3	58	22	14
Karate	4	34	11	7
Football	13	112	46	19
Copperfield	4	115	50	36
Facebook	27	3612	448	656
SN2	11	1999	689.3	814.47

Table 4: Degree Method (Graph Traversed) BFS

It can be observed from the tables 4 and 5 that the degree based Selective Breadth First Traversal algorithm performs better than the one based on clustering, according to the fraction of the network traversed. This has been shown in the Figures 8 and 9. The standard deviation in the case of clustering based SBFT is lesser than that in the case of degree based SBFT.

## 6. CONCLUSION AND FUTURE WORK

The paper proposes two algorithms which take as input two nodes in a network and output 1 if they belong to the same community else 0. The interesting fact about the algorithms is that neither do we explore the entire network, nor do we need to do it. Modified breadth and depth first traversals allow us to do so. It is shown that the algorithms are 87% accurate for the networks having high modularity values and greater than 50% even for the networks having very less modularity values.

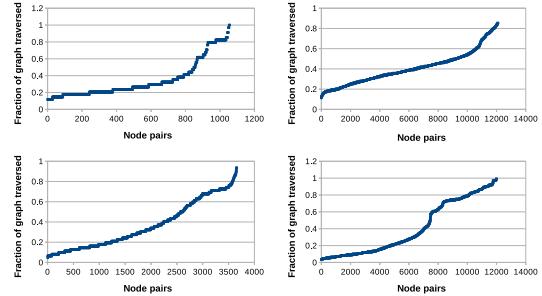


Figure 8: Fraction of graphs traversed according to the degree method. Left to Right: 1) Karate Club Network 2) Football Network 3) Dolphin Network 4) Copperfield Network

Name	Min	Max	Avg	StDev
Dolphin	5	61	32	12
Karate	4	34	18	10
Football	13	110	73	28
Copperfield	4	115	85	29
Facebook	199	3538	557	569
SN2	11	2000	854	818

Table 5: Clustering Method (Graph Traversed) BFS

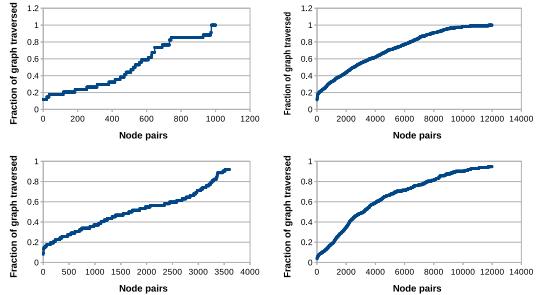


Figure 9: Fraction of graphs traversed according to the clustering method. Left to Right: 1) Karate Club Network 2) Football Network 3) Dolphin Network 4) Copperfield Network

The current work assumes the network to possess non-overlapping communities. But overlapping communities are frequently encountered in real world scenarios. The efficiency of the algorithm can be greatly increased if one considers the communities to be overlapping. Moreover, the algorithm assigns the  $\delta(i, j)$  values a strictly binary number, 0 or 1. One can attempt to assign a fuzzy value to this membership parameter and give the probability with which two nodes might belong to the same community. It would be of greater use specially in the case of overlapping community structure. We have considered only two local parameters, degree and clustering in our work. One might go one step ahead and employ more parameters or hybridised local parameters for the same to improve the results. Nevertheless, according to the best of our knowledge, this question has been first raised in this way for the first time and a lot of interesting ventures in this direction can be pursued.

## 7. REFERENCES

- [1] R. Andersen and K. J. Lang. Communities from seed sets. In *Proceedings of the 15th international conference on World Wide Web*, pages 223–232. ACM, 2006.
- [2] E. R. Barnes. An algorithm for partitioning the nodes of a graph. *SIAM Journal on Algebraic Discrete Methods*, 3(4):541–550, 1982.
- [3] S. Boettcher. 11 extremal optimization. *New optimization algorithms in physics*, page 227, 2004.
- [4] M. Chau and J. Xu. Mining communities and their relationships in blogs: A study of online hate groups. *International Journal of Human-Computer Studies*, 65(1):57–70, 2007.
- [5] J. Chen and B. Yuan. Detecting functional modules in the yeast protein–protein interaction network. *Bioinformatics*, 22(18):2283–2290, 2006.
- [6] P. Chen and S. Redner. Community structure of the physical review citation network. *Journal of Informetrics*, 4(3):278–290, 2010.
- [7] W. Cui, Y. Xiao, H. Wang, and W. Wang. Local search of communities in large graphs. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 991–1002. ACM, 2014.
- [8] P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti. Mixing local and global information for community detection in large networks. *Journal of Computer and System Sciences*, 80(1):72–87, 2014.
- [9] J. Duch and A. Arenas. Community detection in complex networks using extremal optimization. *Physical review E*, 72(2):027104, 2005.
- [10] J. Gao, F. Liang, W. Fan, C. Wang, Y. Sun, and J. Han. On community outliers and their efficient detection in information networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 813–822. ACM, 2010.
- [11] M. Girvan and M. E. Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.
- [12] K.-I. Goh, E. Oh, B. Kahng, and D. Kim. Betweenness centrality correlation in social networks. *Physical Review E*, 67(1):017101, 2003.
- [13] P. Holme, B. J. Kim, C. N. Yoon, and S. K. Han. Attack vulnerability of complex networks. *Physical Review E*, 65(5):056109, 2002.
- [14] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell system technical journal*, 49(2):291–307, 1970.
- [15] J. Kleinberg and S. Lawrence. The structure of the web. *Science*, 294(5548):1849–1850, 2001.
- [16] I. M. Kloumann and J. M. Kleinberg. Community membership identification from small seed sets. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1366–1375. ACM, 2014.
- [17] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical review E*, 78(4):046110, 2008.
- [18] V. Latora and M. Marchiori. Efficient behavior of small-world networks. *Physical review letters*, 87(19):198701, 2001.
- [19] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. <http://snap.stanford.edu/data>, June 2014.
- [20] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Statistical properties of community structure in large social and information networks. In *Proceedings of the 17th international conference on World Wide Web*, pages 695–704. ACM, 2008.
- [21] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.
- [22] W. Lin, X. Kong, P. S. Yu, Q. Wu, Y. Jia, and C. Li. Community detection in incomplete information networks. In *Proceedings of the 21st international conference on World Wide Web*, pages 341–350. ACM, 2012.
- [23] M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, pages 415–444, 2001.

- [24] M. Mitchell, S. Forrest, and J. H. Holland. The royal road for genetic algorithms: Fitness landscapes and ga performance. In *Proceedings of the first european conference on artificial life*, pages 245–254. Cambridge: The MIT Press, 1992.
- [25] M. E. Newman. Fast algorithm for detecting community structure in networks. *Physical review E*, 69(6):066133, 2004.
- [26] M. E. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3):036104, 2006.
- [27] A. Pothen. Graph partitioning algorithms with applications to scientific computing. In *Parallel Numerical Algorithms*, pages 323–368. Springer, 1997.
- [28] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9):2658–2663, 2004.
- [29] T. Richardson, P. J. Mucha, and M. A. Porter. Spectral tripartitioning of networks. *Physical Review E*, 80(3):036111, 2009.
- [30] P. R. Suaris and G. Kedem. An algorithm for quadrisection and its application to standard cell placement. *IEEE Transactions on Circuits and Systems*, 35(3):294–303, 1988.
- [31] Y. Sun, B. Danila, K. Josić, and K. E. Bassler. Improved community structure detection using a modified fine-tuning strategy. *EPL (Europhysics Letters)*, 86(2):28004, 2009.
- [32] M. Tasgin, A. Herdagdelen, and H. Bingol. Community detection in complex networks using genetic algorithms. *arXiv preprint arXiv:0711.0491*, 2007.
- [33] J. R. Tyler, D. M. Wilkinson, and B. A. Huberman. E-mail as spectroscopy: Automated discovery of community structure within organizations. *The Information Society*, 21(2):143–153, 2005.
- [34] G. W. Tyson, J. Chapman, P. Hugenholtz, E. E. Allen, R. J. Ram, P. M. Richardson, V. V. Solovyev, E. M. Rubin, D. S. Rokhsar, and J. F. Banfield. Community structure and metabolism through reconstruction of microbial genomes from the environment. *Nature*, 428(6978):37–43, 2004.
- [35] I. Vragović and E. Louis. Network community structure and loop coefficient method. *Physical Review E*, 74(1):016105, 2006.
- [36] L. Weng, F. Menczer, and Y.-Y. Ahn. Virality prediction and community structure in social networks. *Scientific reports*, 3, 2013.
- [37] D. R. White and K. P. Reitz. Graph and semigroup homomorphisms on networks of relations. *Social Networks*, 5(2):193–234, 1983.
- [38] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, pages 452–473, 1977.

# Minimizing Network Traffic Features for Android Mobile Malware Detection

Anshul Arora

Department of Computer Science and  
Engineering  
Indian Institute of Technology Roorkee  
Roorkee, India  
ansh7.dcs2014@iitr.ac.in

## ABSTRACT

Smartphones have emerged as one of the dominant computing platforms in today's era where Android has been the first choice for users as well as app developers due to its open source nature and feature rich apps. Such popularity has come hand-in-hand with an equivalent increase in malware targeting Android. Since mobile devices allow easy-to-use, touch-sensitive, and anywhere-anytime access to its resources, the mobile-specific applications like SMS, MMS, Bluetooth, e-mail, and other services may pose serious threats and lead to financial losses and privacy leakages. In recent time, high attention is drawn by the researchers for detecting Android malware; very fewer community have considered network traffic analysis in their detection models. The majority of these models have considered the detection primarily on traffic features that distinguish malware traffic from normal traffic. In this paper, we have proposed an algorithm to prioritize network traffic features with an aim to minimize the number of features to be analyzed to give high detection accuracy along with reduced training and testing time. To this extent, we have used statistical tests to rank the features. Results demonstrate that using prioritized features for detection not only reduces the training and testing time, but also gives slightly higher detection accuracy than using all the features together by measuring  $F_{measure}$ , a widely used measure for detection accuracy. The training time of 300 applications is reduced from 11.7 seconds to 5.8 seconds and testing time of 230 applications is reduced from 25.1 seconds to 17.3 seconds, hence reduction of around 50% and 31% in training time and testing time respectively. We believe this time difference will have a larger impact if there are thousands of files to be tested.

## Keywords

Android, Smartphone, Malware, Network Traffic, Feature Analysis, Feature Ranking, Malware Detection, Mobile Malware

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

*ICDCN '17, January 04-07, 2017, Hyderabad, India*

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007763>

Sateesh K Peddoju

Department of Computer Science and  
Engineering  
Indian Institute of Technology Roorkee  
Roorkee, India  
[drpskfec@iitr.ac.in](mailto:drpskfec@iitr.ac.in)

## 1. INTRODUCTION

Smartphones are becoming more powerful than early Personal Computers (PCs) because of their fascinating computing capabilities. There has been a tremendous growth in the sales of smartphones and is expected to reach around 2.6 billion by the year 2019 [1]. Google's Android has beaten all its competitors by a huge margin of nearly 80% [2]. Attackers always target on weak systems and try to attack on typically huge masses. Since 2010 Android has been the top-most target for malware developers as more than 90% of the mobile malware are targeted towards Android platform and the number is increasing everyday. Techniques like repackaging, update attacks and drive-by-downloads are used by attackers to inject malicious payloads into normal apps and app markets provide an easy channel for malicious samples to propagate to users. Threats posed by malware include financial loss, privacy leakage, root exploits and mobile botnets [3]. Since Android is the most targeted mobile platform for malware developers, so we have used Android mobile platform for malware detection rather than any other like iOS or Windows.

There are many mobile malware that are remotely controlled as they receive commands from the server and may leak private information of the user or device to the server. There are some other malware that initially do not contain any malicious code but after installation on the device they inject malicious code during the update. Both of these categories of malware have one thing in common i.e. connecting to a network. According to [3] around 93% of Android malware samples have network connectivity. Since these type of malware are controlled by a remote server, it converts the mobile device into a mobile bot which can pose serious threat to user community.

Although network traffic has been widely used in computer based intrusion detection but it has not been explored much in mobile malware detection. Network traffic features for mobile malware detection have been used by very few works as compared to the other features like permissions, java code or system calls. Out of those studies, most of the works like [4] have used Android emulator for capturing network traffic of malware samples. The limitation with emulator based studies is that they do not support events like sending an SMS, dialing a number or rebooting of phone etc. Therefore malware that wait for such events may not trigger their malicious payload and as a consequence may not communicate with the remote server, producing no or very low amount of network traffic. So with emulator we

may get traffic of some of the remotely controlled malware but not all. Works like [13], [14] have focused on finding deviations in network traffic of malicious samples and normal samples. But none of them, to the best of our knowledge, have focused on finding the most important set of features that can be used in detecting Android malware samples using network traffic.

In this paper we aimed at analyzing network traffic behavior of Android malware by capturing their traffic from actual smartphones, not the emulator and prioritized the traffic features so that a subset of features is used for detection rather than whole set of features. The benefits of such prioritization of features are: (a) reduction of features that may have, otherwise, negative impact on the detection performance; and (b) fast detection of malicious network activity. Although traffic features have been analyzed in our previous work [4] in which we found that seven network traffic features can distinguish between normal traffic and malicious traffic; but the traffic was captured from an emulator. As discussed earlier, there are limitations in using the emulator for capturing network traffic of Android malware. Therefore we used actual smartphones to capture the network traffic. Results have shown quite significant deviations in their network traffic as compared to those that were captured from the emulator. Moreover no single feature can distinguish between malware and normal traffic as compared to the previous work. Since no distinguishing features exist, we need to design an algorithm to find the minimal set of features that can give maximum detection accuracy. In this work statistical techniques are used to rank the set of 22 features. A unique algorithm is designed to extract the top features from different rankings and results show that 9 features out of total of 22 are sufficient to give maximum detection accuracy. These 9 features not only give higher detection rate but also save considerable amount of time in terms of training and testing the data-set. Training time is reduced by almost 50% and testing time is reduced by around 31% using the prioritized set of features. The contributions of the paper can be summarized as:

1. Actual smartphones are used in the experiments rather than emulator to capture Android malware network traffic, enabling to capture real time traffic when malware are communicating with their remote server.
2. Prioritized network traffic features for Android malware to determine features that are highly important for detecting malicious network activity in smartphones.
3. An algorithm is designed to extract important features based upon the rankings obtained by statistical techniques.
4. Validated through experiments that Android malware can be detected with smaller set of features with high accuracy and less execution time.

The rest of the paper is organized as follows: Section 2 discusses the related work done in Android malware detection with focus on techniques using traffic features for analysis and detection. Proposed Methodology is presented in section 3. Results are discussed in section 4 and section 5 concludes the paper with future work directions.

## 2. RELATED WORK

Techniques proposed for Android malware detection fall in either of two categories: Static Analysis and Dynamic Analysis. Static Analysis deals with finding dangerous permissions in the manifest file or risky keywords within the source code of the app without executing the application. Enck *et al* [5] developed Kiran, one of the earliest on-device defense solution which evaluated risks involved with Android applications in terms of dangerous permissions combination and generated the set of security rules based upon them. Droid-Analyzer [6], another static tool was designed to find root privileges within the Android malware. It first extracted the list of risky keywords associated with malware samples and then searched for these keywords in their testing data set of apps for banking or flight booking. Permissions pattern algorithm [7] found top 20 permissions used by malicious as well as normal apps. They used both "requested permissions" and "used permissions" to find the distinguishing permission patterns of malware and normal applications. In [8] manifest file components like intents, activities, services and broadcast receivers were considered and inter component communication was analyzed between them. Machine learning algorithms like SVM, Decision Trees and Random Forests were used as detection model. Measures like mutual information, T-Test and Correlation Coefficient were used in [9] to determine the top permissions in malicious samples. Techniques like Sequential Forward Selection and Principal Component Analysis were used to identify risky permission sets. All such static solutions are inexpensive as their code or manifest files are analyzed to give the decision without executing the application. But such techniques lack in detecting stealthy malware that download their malicious payload at run time. For example malware that have update attack capability which download malicious component along with their updates, go undetected and therefore static solutions cannot detect such malware samples.

Dynamic Techniques are proposed to overcome limitations of static techniques in which run time suspicious behavior of applications is observed by executing the application on actual mobile device or on emulator. Features like system calls and network traffic have been used in monitoring run time behavior of applications. An on-device detection model named *TaintDroid* was developed in [10] to prevent information leakage from the mobile device. The model labeled the sensitive data and recorded the destination address whenever the sensitive data leaves the system. Data was labeled (tainted) at four different levels: file level, method level, message level and variable level. Burgera *et al* [11] collected logs of system calls for each application and applied clustering to distinguish between normal system calls and the malicious ones. SCSDroid [12], another dynamic tool aimed to detect repackaged apps from the third party app markets without the need of original application. The authors extracted thread grained system call sequences from each application rather than process grained system calls because they believed that malicious behavior can be observed in a single thread of the process application running on Android emulator.

Although network traffic based detection techniques are suitable for a subset of total malware samples due to the fact that not all malware need network connectivity but still it can be applied for 93% of the Android samples. Shabtai *et al* [13] focused on detecting malware with self updating

capabilities by analyzing deviations in their traffic behavior. Features like average bytes sent, average bytes received, interval between packets sent and interval between packets received etc. were analyzed. Decision Trees and RepTree algorithms were used for detecting anomalies in traffic behavior. The experiments were performed on 5 real malware and 10 self written malicious codes. In our recent work we found seven features which can distinguish malware traffic from normal traffic and rule based classifier was built based upon those seven features [4]. Chen *et al* [14] investigated Android malware traffic and features like DNS queries, HTTP packet length, HTTP requests, ratio of downlink to uplink traffic and Ad traffic were analyzed. Their study found that more than 70% of malware generate malicious traffic within first five minutes and other features can identify malware traffic. Li *et al* [15] examined features like Source IP, Destination IP, Source Port, Destination Port , Upward and Downward flow for malware traffic and applied SVM algorithm for detection. Feizollah *et al* [16] evaluated k-means and mini batch k-means algorithms on network traffic of Android malware. They analyzed features like frame length, frame number, connection duration, source and destination port numbers and found mini batch k-means is better than k-means. Zaman *et al* [17] captured Android malware traffic and recorded the URI each application communicated with and detection was made with the help of domain blacklist. Network traffic has also been used in two other studies for different mobile OS malware detection like Symbian malware and iOS malware [18][19]. Features like Byte sequence, IP blacklisting, connection success ratio, flow duration and number of common destinations have been used for detecting malicious activities in mobile devices.

At the best of our knowledge none of the existing work has focused on prioritizing network traffic features for Android malware so that fewer set of features can be analyzed for detection. Using important features for malware detection not only increases accuracy but also helps in reducing the training and testing time. In our previous work [4] we have found seven distinguishing features but traffic was captured from emulator. In this work when actual smartphones are used, there are significant deviations in network traffic of malware samples and hence no distinguishing feature has been found. So aim of this work is to prioritize the features such that a subset of total features should be analyzed together for effective Android malware detection. Moreover we use larger set of features, a total of 22, in this work as compared to total of 16 features used in the previous work.

### 3. METHODOLOGY

In our earlier work [4] malware and normal mobile traffic was captured on Android emulator, but this has a serious limitation that many malware samples wait for some system related events like reboot of phone, receiving or dialing a phone call, receiving or sending a SMS etc. Since such events are not feasible in case of emulator, these samples cannot trigger their malicious payload on the smartphone and as a result may not properly communicate with its remote server. So they produce no or very less network traffic. However in this work, actual smartphones are used for capturing their network behavior. Our previous work uses 16 set of features out of which we found 7 of them as distinguishing features. For example Ratio of Incoming to Outgoing Bytes for malware traffic was in the range of 0.1 - 3.9 and

for normal traffic its value was between 6 - 7. Since the range for malware and normal traffic is non-overlapping, we call this feature as distinguishing feature. We extend the total number of features in this work to 22, values of all these features for malware traffic and normal mobile traffic are given in Table 1. These features are coded in second column of the table for simplicity and will be referred by these notations in the rest of the paper. List of Android malware that are used in experiments is given in Table 2. The complete list is divided into two sets: one set (set I) is used for training. This data set is used for testing the detection model using prioritized features. Second set (set II) of samples are used as unknown samples since these are not used for training purpose. After the prioritized set of features is extracted from 22 features, we tested the effectiveness of these features on unknown samples. Therefore we kept these samples separated from the samples in set I.

Samsung Galaxy S4 smartphone running Android version 4.2 is used to capture network traces of both malware as well as normal applications. Each sample was allowed to run for the period of 72 hours. Normal traffic includes traffic of social networking applications like Whatsapp, Facebook, Twitter, LinkedIn etc; online gaming applications and normal mobile web browsing on the mobile Internet. Total of 105 normal network traces are used in training and 80 traffic files are used in testing. Mobile device is continuously connected to Internet through Wi-Fi. The traffic in pcap format can be easily analyzed by software like Wireshark. Each traffic file is transmitted to the desktop system running Ubuntu 12.04 OS with 8 GB RAM for further analysis.

It can be clearly analyzed from the Table 1 that no feature is distinguishable. Since no such distinguishing features exist, determining the minimal set of features among them is the big challenge. To determine the best feature set, the features should be ranked. In the next subsection we discuss feature ranking techniques used.

### 3.1 Feature Ranking Methods

The purpose of feature ranking is multi fold. Irrelevant features may not contribute much in detection and may even reduce the detection accuracy. Therefore prioritizing the features can improve the detection accuracy. Secondly, reduced feature dimensionality results in lower set of features to be used in training and testing that can reduce training and testing time of the detection algorithm. We use Information Gain (I.G.) and statistical technique like Chi-Square Test (CHI) to determine the ranking of the 22 features.

#### 3.1.1 Information Gain

Information Gain is used to determine which feature in a given set of feature vectors is most useful in discriminating the classes to be learned. Entropy characterizes the impurity or measures the uncertainty of attributes. Information gain is the measure of reduction of uncertainty. Suppose there is a training set  $T$  containing features  $F_1, F_2.. F_n$  and a class label  $C$ . The entropy or uncertainty for class  $C$  is defined as :

$$H(C) = - \sum_i P(C_i) \log_2(P(C_i)) \quad (1)$$

where  $P(C_i)$  represents the prior probability for all values of  $C$ . The uncertainty for class  $C$  after observing feature  $F$  is given by the conditional entropy :

**Table 1: Traffic Features and their range for malware and normal mobile traffic on smartphones**

Traffic Feature	Feature Notation	Range in malware traffic	Range in normal traffic
Average Packet Size	F1	72-388	162-2324
Ratio of Incoming to Outgoing Bytes	F2	0.01-6.013	1.35-55
Bytes Received per Second	F3	3.8-25555.6	1015-33895
Packets Sent per Flow	F4	1.43-396	1.5-118
Packets Received per Flow	F5	1 - 30.245	6.1-50.8
Bytes Sent per Flow	F6	111-409485	117-7054
Bytes Received per Flow	F7	1.1-166460	62-659717
Bytes Sent per Second	F8	8.4-79433.1	3-40311
Packets sent per Second	F9	0.06-1470.64	0.04-487
Packets Received per Second	F10	0.05-1470.62	0.01-486
Average Time Interval b/w Packets Sent	F11	0.09-2745.23	0.8-98
Average Time Interval b/w Packets Received	F12	0.01-3234.22	0.9-123
Ratio of Incoming to Outgoing Packets	F13	0.01-0.99	0.4-0.98
Average Flow Duration	F14	0.6-9715.29	9-492
Ratio of Number of Connections to Number of Destination IPs	F15	1-4093	1-8
Maximum Packet Size	F16	78-1935	75-17726
First Packet Size Sent	F17	69-1070	54-423
First Packet Size Received	F18	66-265	54-4315
Minimum Time Interval b/w Packets Sent	F19	0-3.26	0-6.07
Maximum Time Interval b/w Packets Sent	F20	0.3-17900	6-2066
Minimum Time Interval b/w Packets Received	F21	0 - 5.31	0-10.135
Maximum Time Interval b/w Packets Received	F22	0.4-48721	10.5-2582.91

$$H(C|F) = - \sum_j P(F_j) \sum_i P(C_i|F_j) \log_2(P(C_i|F_j)) \quad (2)$$

where  $P(C_i|F_j)$  represents the posterior probabilities of class  $C$  given the value of feature  $F$ . Now the information gain of feature  $F$  with respect to class  $C$  can be given as:

$$IG(C|F) = H(C) - H(C|F) \quad (3)$$

To evaluate the information gain for any feature, we need to combine its values of malware as well as normal traffic and check at which values, the class label is changing. If this class label is changing very frequently, then its information gain will be low because it is difficult to classify unknown samples from this feature. For any two features  $F_1$  and  $F_2$ , we calculate  $IG(C|F_1)$  and  $IG(C|F_2)$  respectively. Higher information gain of any feature say  $F_1$  reflects the fact that  $F_1$  can classify unknown samples correctly with higher probability than other features. Therefore the feature having more information gain will be ranked higher and equation 3 is used to determine rank of all the 22 features.

### 3.1.2 Chi-Square Test

Chi-Square Test measures the difference between the expected values and the observed values and determines whether the deviation between observed and expected value is acceptable or not. Chi-Square value is calculated by the equation (4):

$$\chi^2 = \sum_{k=1}^n \frac{(O_k - E_k)^2}{E_k} \quad (4)$$

where  $O_k$  and  $E_k$  represent observed value and expected value respectively for any sample  $k$  in the experiment. Lesser the deviation between the observed and the expected value, lower is the value obtained by Chi-Square Test. In this work we have 22 network traffic features and their values for both malware and normal traffic that are kept separately. This test is applied separately on malware traffic and normal traffic, unlike information gain which is calculated by combining both types of traffic. For any particular feature say  $F_i$  we have its range for malware samples. From these values we calculate the average of the feature value to serve as the expected value for our experiments. Then we measure how the values for that feature deviate from the expected value by using the equation 4. The procedure is repeated for normal traffic as well for all features. More closer the values are to the average value, lower is the Chi-Square result and higher is the priority of the feature. So unlike information gain where higher gain leads to high ranking of the feature, lower Chi-Square value leads to higher ranking of the feature.

## 3.2 Feature Selection

The selection of important features among the ranked features is obtained from I.G. and CHI tests and is presented in algorithm 1. We will have three different rankings of features; one from I.G. and two from CHI, viz. CHI applied separately on malware traffic and normal traffic. Three different ranked features is given as an input to determine the minimal set of top ranked features from all three rankings. Our aim is to prioritize the features in such a way that few features should be used for Android malware detection rather than whole set of 22 features. We create two lists: Initial list ( $I_{list}$ ) and output list ( $O_{list}$ ).  $I_{list}$  contains the common features from all three rankings whereas  $O_{list}$

**Table 2: List of Android malware used in experiments**

Malware	Set I		Malware	Set II	
	# used for training	# used for testing		# used for testing	# used for testing
AnserverBot	30	25	ADRD	20	
BaseBridge	30	25	GoldDream	40	
BgServ	5	4	jSMSHider	15	
DroidDream	10	4	RogueLemon	2	
DroidDreamLight	20	20	DroidKungFu4	24	
DroidKungFu1	20	10	DroidKungFu5	1	
DroidKungFu2	20	10			
DroidKungFu3	30	28			
Pjapps	25	20			
Plankton	8	4			
Total	198	150			102

contains the minimal set of features. We measure detection results in terms of  $F_{measure}$  score which is a measure of detection algorithm's accuracy and is represented in terms of precision and recall; given by the equation 5.

$$F_{measure} = \frac{2 \times (Precision \times Recall)}{Precision + Recall} \quad (5)$$

where Precision and Recall are defined as:

$$Precision = \frac{T_p}{T_p + F_p} \quad (6)$$

$$Recall = \frac{T_p}{T_p + F_n} \quad (7)$$

Here  $T_p$ ,  $F_p$  and  $F_n$  are the number of true positives, false positives and false negatives respectively.

This paper aims to find the maximum  $F_{measure}$  score from minimal set of features used. As presented in algorithm 1, for every value of  $k$ , we find the common features from the top  $k$  ranked features of all three lists.  $(F_{list1})_k$ ,  $(F_{list2})_k$  and  $(F_{list3})_k$  represents the top  $k$  features from rankings of I.G., CHI test on malware samples and CHI test on normal samples respectively. The common features from all three rankings are listed in a temporary list  $temp$  and finally this set is added to  $I_{list}$ . Duplicate elements are deleted in  $I_{list}$ . This step is needed because as  $k$  increments,  $temp$  will contain the common features from the top  $k$  ranked features of all the lists containing features that are already added to the  $I_{list}$ . If we get common features at any iteration of  $k$ , then  $I_{list}$  will be updated with those new common features. Then Naïve Bayes' detection algorithm is used for detection of test data using the features given in  $I_{list}$ . Suppose  $k$  features are there in  $I_{list}$ ; so those  $k$  features will be used for testing. Those  $k$  features are extracted for every sample to be tested. Our dataset consists of numerical attributes rather than the categorical ones. We have applied Naïve Bayes' on numerical attributes assuming the numerical data follows normal distribution. Probability Density Function (PDF) is calculated which acts as a priori estimate to be used in Naïve Bayes'.  $F_{measure}$  score is determined based on the results and if this score is better than the previous one, then the features from  $I_{list}$  are assigned to  $O_{list}$ . It may happen that on adding a new feature in  $I_{list}$ , detection

algorithm gives same  $F_{measure}$  score as the previous one. In this case we will not add the new features in  $O_{list}$  because we are interested in finding the minimal set of features which gives us the best detection results. Once all the iterations are over the algorithm returns the  $O_{list}$  set which contains the minimal set of features with highest F-score.

---

**Algorithm 1** Selection of important features from ranked features

---

**Input:** Three rankings of features:  $F_{list1}$ ,  $F_{list2}$  and  $F_{list3}$  obtained from I.G., CHI on malware and CHI on normal respectively  
**Output:** Minimal set of features which gives best detection results

```

1:  $I_{list} \leftarrow \emptyset$  ,  $O_{list} \leftarrow \emptyset$  ,  $F_{max} = 0$ 
2: for  $k = 1 \rightarrow 22$  do
3:    $temp \leftarrow (F_{list1})_k \cap (F_{list2})_k \cap (F_{list3})_k$ 
4:    $I_{list} = I_{list} \cup (I_{list} - temp)$ 
5:   Run Naïve Bayes' classifier on test data using features in  $I_{list}$ 
6:   Calculate  $F_{measure}$  score  $F_{score}$  of detection results
7:   if  $F_{score} > F_{max}$  then
8:      $F_{max} \leftarrow F_{score}$ 
9:      $O_{list} \leftarrow I_{list}$ 
10:  end if
11: end for
12: Return  $O_{list}$ 

```

---

## 4. RESULTS AND DISCUSSION

Section 4.1 discuss the rankings obtained by the features by applying I.G. and CHI tests on the whole set of 22 features. The detection results are also described in this section where we show using prioritized features give higher detection rate than using all of the features. Effectiveness of prioritized features on set of unknown malware traffic is discussed in section 4.2 which shows unknown samples with prioritized features can be detected with same accuracy as with all features except for very few exceptions. Section 4.3 shows that the prioritized set obtained by algorithm 1 that considers all three rankings is more suitable set with higher detection accuracy as compared to other sets obtained by features from individual rankings. In the next section, we show what amount of time is saved by using important set

of features for training and testing. And last subsection discusses the limitations of this approach.

## 4.1 Ranking of Traffic Features

Ranking of features according to I.G. and CHI test for all 22 features considered in this work is given in the Table 3. I.G. is calculated by considering both the malware and normal values of any particular traffic feature. CHI test for every feature on the other hand is done separately on malware and normal samples. Feature  $F_{13}$  i.e Ratio of Incoming to Outgoing Packets, produces the lowest CHI value because in both malware and normal traffic, value of this feature for all samples lies close to the average value of the feature, hence giving the lowest CHI value and the highest priority. On the other hand range of this feature for malware and normal traffic is quite similar: 0.01 - 0.99 for malware traffic and 0.4-0.98 for normal traffic. Since the range is very much similar, hence I.G. for this feature is the least hence giving it the lowest priority. So there may be the case when any feature gets the highest priority in one test and lowest in other. Therefore we need to choose the set of features from both tests which can give the best detection results.

Applying our feature selection algorithm on ranked features from both tests, we get the results as shown in Table 4. The first column shows the value of  $k$  i.e common features from the ranked lists by considering top  $k$  features from all three rankings. Till  $k = 3$  there is no single common feature from all three lists as can be seen in Table 3. Second column ( $I_{list}$ ) lists the common features selected from the rankings from top  $k$  features of each ranked list. Third column named Final List of Features ( $O_{list}$ ) contains the minimal set of features that produces better  $F_{measure}$  score than the previous score. This column will give us the final output of the prioritized features which contribute the most in detection. Even on adding new features to the second column, we are getting the same detection results and same  $F_{measure}$  score as can be seen in the iterations from  $k = 13$  to 21. So we do not consider those features in our final list of output as we want to find the minimal set of features which contribute the most in detection. Features are added in the final list only if  $F_{measure}$  score is better than the previous score.

At  $k = 13$  we get the highest  $F_{measure}$  score considering nine traffic features  $\{F_2, F_{19}, F_{21}, F_1, F_5, F_{16}, F_4, F_{10}, F_9\}$  as given in third column of table. It correctly detects 146 malicious samples and 73 normal samples out of total of 150 and 80 respectively. Adding more features to this list produces the same detection results till the last iteration in which features like  $F_7$  and  $F_8$  are added which are ranked higher in information gain but lowest in chi square test. On addition of these features, 147 malicious samples (1 more results than from 9 features) and 71 normal samples (2 less results than from 9 features) are detected correctly, giving the  $F_{measure}$  score of 0.9607 which is smaller than  $F_{measure}$  score of 0.9636 obtained from 9 features. Therefore we conclude that set of 9 features  $\{F_2, F_{19}, F_{21}, F_1, F_5, F_{16}, F_4, F_{10} \text{ and } F_9\}$  obtained at 13th iteration is the minimal set of features that gives us the higher detection rate rather than considering all 22 features for detection.

## 4.2 Unknown Samples

To check the effectiveness of our proposed work on unknown samples we kept some malicious samples like ADRD, RogueLemon, GoldDream, jSMSHider, DroidKungFu4 and

DroidKungFu5 separated from the training as well as the testing data-set. These samples are not included in the training database while we extracted features from the network traces and hence they are termed as unknown. Total of 102 samples as given in Table 2 are tested using the set of nine features and then using all 22 features. The detection results are shown in Table 5. It can be seen that for all malware families except GoldDream and DroidKungFu4, Naïve Bayes' classifier gives equal detection accuracy with both: 9 set of features and 22 set of features. The difference in detection results is due to the fact that the 4 samples (2 of GoldDream and 2 of DroidKungFu4) generate large amount of network traffic as compared to other malicious network traces. GoldDream malware tries to upload phone call related information like duration of the call, outgoing or incoming number of the call etc. and SMS related information like SMS contents to the remote server [3]. While those two samples of GoldDream are being captured as compared to other captures, there was heavy phone related activity on mobile device which GoldDream may have tried to upload to its server. The nine prioritized features are not able to detect such activity as malicious because features like Ratio of Incoming to Outgoing Bytes ( $F_2$ ) and others varied close to the range of normal traffic. Except for these four samples, detection accuracy using nine features is equivalent to that of using 22 features. So we conclude that 9 set of features can detect unknown samples as well with good detection accuracy.

## 4.3 Minimal Feature Set

In this section we discuss whether the minimal set obtained from 9 features is the suitable set that can give high detection accuracy. We argue whether less than 9 features or some other combination of 9 features can give better detection results or not. To test this argument we make three other sets of nine features each of them are extracted from three different rankings.  $Set_1$  is the set of top 9 features obtained from information gain rankings i.e.  $Set_1 = \{F_{16}, F_{21}, F_2, F_{19}, F_1, F_7, F_3, F_8, F_5\}$ .  $Set_2$  is the set of top 9 features obtained from Chi-Square test on malware traffic i.e.  $Set_2 = \{F_{13}, F_2, F_{19}, F_{21}, F_5, F_{18}, F_1, F_4, F_{16}\}$ .  $Set_3$  is the set of top 9 features obtained from Chi-Square test on normal traffic i.e.  $Set_3 = \{F_{13}, F_{15}, F_{19}, F_2, F_{21}, F_5, F_4, F_1, F_{12}\}$ . The prioritized feature set obtained from the proposed algorithm is say  $Set_4$  is  $\{F_2, F_{19}, F_{21}, F_1, F_5, F_{16}, F_4, F_{10}, F_9\}$ . We compare the detection results from these 4 different sets to check which set has the higher detection accuracy. Figure 1 and Figure 2 compare the number of true positives and false positives for all 4 sets. The false negatives and true negatives are complement of true positives and false positives respectively. So we have drawn figures of true positives and false positives only. For every value on x-axis from 1 to 9; which represent the number of features, y-axis shows the number of true positives obtained from the corresponding feature(s) used for detection. As we increment the number of features to be used for detection from 1 to 9, as expected number of true positives also increase for every set. But as we reach to the set of 9 features,  $Set_4$  gives the maximum number of true positives as compared to other sets.  $Set_4$  is able to detect 146 samples correctly as compared to 142 from  $Set_1$ , 141 from  $Set_2$  and 130 from  $Set_3$ . Same is the case with number of false positives;  $Set_4$  generates 9 false positives as compared to 15 from  $Set_1$ , 25 from

**Table 3: Ranking of Features according to Information Gain and Chi Square Tests**

Rank according to Information Gain	Rank according to Chi Square Test	
	On Malware Samples	On Normal Samples
F16	F13	F13
F21	F2	F15
F2	F19	F19
F19	F21	F2
F1	F5	F21
F7	F18	F5
F3	F1	F4
F8	F4	F1
F5	F16	F12
F9	F17	F16
F10	F10	F14
F4	F11	F10
F15	F9	F9
F14	F12	F11
F6	F15	F20
F11	F14	F22
F20	F20	F6
F22	F3	F18
F12	F22	F17
F17	F6	F3
F18	F7	F8
F13	F8	F7

$Set_2$  and 27 from  $Set_3$ . So it can be concluded that  $Set_4$  with 9 features obtained from proposed feature selection algorithm from different rankings is better than other sets of same size obtained from individual rankings of features.

Next we need to show that this is the suitable set with minimal features. i.e. no other set with less than 9 features can produce better detection results. Again analyzing figures 1 and 2, we find that when number of features considered are 5 from  $Set_1$  and  $Set_4$ , then  $Set_1$  detects higher number of true positives and lower number of false positives as compared to  $Set_4$ . Number of true positives from  $Set_1$  and  $Set_4$  are 135 and 133 respectively. And number of false positives from  $Set_1$  and  $Set_4$  are 16 and 23 respectively. So we can argue that 5 top most features from information gain ranking i.e. {F16, F21, F2, F19, F1} giving  $F_{measure}$  score of 0.8969 is the minimal set which gives good detection results as compared to top 5 features from  $Set_4$  i.e. {F2, F19, F21, F1, F5} giving  $F_{measure}$  score of 0.8692. But our aim is to give the best detection results i.e. the set which gives highest possible  $F_{measure}$  score which is 0.9636 obtained by 9 features of  $Set_4$ . There can be different discussions on this point that whether using lesser number of features(5) with low F-score is better or using more number of features(9) with higher F-score is better. But we assume that in the field of security, our aim should be that no malware should go undetected. So keeping this in mind we believe that using 9 features with high F-score is better than using 5 features with low F-score. Therefore we conclude that this is the minimal subset and no other set of features with less than 9 features can give better detection results.

#### 4.4 Execution Time

In this section we compare the running times (including both training time and testing time) when features are pri-

oritized and classifier is tested using 9 features and 22 features. It is generally expected that using lesser number of prioritized features for detection should be faster than using all of the features. So such comparison is necessary to show whether actually we have saved some amount of time or not. So we have compared training time and testing time for both cases : using 9 features and using all 22 features. Figures 3 and 4 show the comparison for training time and testing time respectively. As expected using 9 features takes less time (5.8 seconds for training 300 apps traffic and 17.3 seconds for testing 150 apps traffic) as compared to using 22 features (11.7 seconds for training 300 apps traffic and 25.1 seconds for testing 150 apps traffic). We have assumed the training time in this work to be the time required to extract the features from the captured traffic file. And testing time is the sum of time used to extract the features from the traffic file and run Naïve Bayes' classifier to give the detection result for that file. Therefore testing time is higher than the training time in our case. We trained on 303 samples and tested on total of 230 samples (including both normal and malicious). So training time is reduced by around 50% and testing time is reduced by around 31% when using nine prioritized features. This time difference can have a larger impact if we have thousands of files to train and test upon. We have also compared training and testing time of using top five features obtained from  $Set_1 = \{F16, F21, F2, F19, F1\}$  in the previous section. Using these five features take around 4.7 seconds for training 303 apps and 13 seconds for testing 230 apps, reduction of around 18% and 24% respectively from that of nine features. So time reduction from 22 features to nine features was more than from nine features to five features. Moreover nine features give higher detection rate, therefore we conclude this set of nine features is

**Table 4: Detection results**

Value of k	Common Features from top k features from three rankings ( $I_{list}$ )	Final List of Features ( $O_{list}$ )	TPR%	FPR%	Precision	Recall	$F_{measure}$
1,2,3	-	-	-	-	-	-	-
4	F2, F19	F2, F19	80%	37.5%	0.8	0.8	0.8
5	F2, F19, F21	F2, F19, F21	85.3%	32.5%	0.8311	0.8533	0.8421
6	F2, F19, F21	F2, F19, F21	85.3%	32.5%	0.8311	0.8533	0.8421
7	F2, F19, F21	F2, F19, F21	85.3%	32.5%	0.8311	0.8533	0.8421
8	F2, F19, F21, F1	F2, F19, F21, F1	87.3%	30%	0.84516	0.87333	0.85901
9	F2, F19, F21, F1, F5	F2, F19, F21, F1, F5	88.6%	28.75%	0.85256	0.88666	0.8692
10	F2, F19, F21, F1, F5, F16	F2, F19, F21, F1, F5, F16	96%	11.25%	0.9411	0.96	0.9504
11	F2, F19, F21, F1, F5, F16	F2, F19, F21, F1, F5, F16	96%	11.25%	0.9411	0.96	0.9504
12	F2, F19, F21, F1, F5, F16, F4, F10	F2, F19, F21, F1, F5, F16, F4, F10	96.6%	11.25%	0.9415	0.9666	0.9538
13	<b>F2, F19, F21, F1, F5, F16, F4, F10, F9</b>	<b>F2, F19, F21, F1, F5, F16, F4, F10, F9</b>	<b>97.3%</b>	<b>8.75%</b>	<b>0.9542</b>	<b>0.9733</b>	<b>0.9636</b>
14	F2, F19, F21, F1, F5, F16, F4, F10, F9	F2, F19, F21, F1, F5, F16, F4, F10, F9	97.3%	8.75%	0.9542	0.9733	0.9636
15	F2, F19, F21, F1, F5, F16, F4, F10, F9, F15	F2, F19, F21, F1, F5, F16, F4, F10, F9	97.3%	8.75%	0.9542	0.9733	0.9636
16	F2, F19, F21, F1, F5, F16, F4, F10, F9, F15, F11, F14	F2, F19, F21, F1, F5, F16, F4, F10, F9	97.3%	8.75%	0.9542	0.9733	0.9636
17	F2, F19, F21, F1, F5, F16, F4, F10, F9, F15, F11, F14, F20	F2, F19, F21, F1, F5, F16, F4, F10, F9	97.3%	8.75%	0.9542	0.9733	0.9636
18	F2, F19, F21, F1, F5, F16, F4, F10, F9, F15, F11, F14, F20	F2, F19, F21, F1, F5, F16, F4, F10, F9	97.3%	8.75%	0.9542	0.9733	0.9636
19	F2, F19, F21, F1, F5, F16, F4, F10, F9, F15, F11, F14, F20, F12, F22	F2, F19, F21, F1, F5, F16, F4, F10, F9	97.3%	8.75%	0.9542	0.9733	0.9636
20	F2, F19, F21, F1, F5, F16, F4, F10, F9, F15, F11, F14, F20, F12, F22, F17, F3, F6	F2, F19, F21, F1, F5, F16, F4, F10, F9	97.3%	8.75%	0.9542	0.9733	0.9636
21	F2, F19, F21, F1, F5, F16, F4, F10, F9, F15, F11, F14, F20, F12, F22, F17, F3, F6, F18	F2, F19, F21, F1, F5, F16, F4, F10, F9	97.3%	8.75%	0.9542	0.9733	0.9636
22	F2, F19, F21, F1, F5, F16, F4, F10, F9, F15, F11, F14, F20, F12, F22, F17, F3, F6, F18, F7, F8, F13	F2, F19, F21, F1, F5, F16, F4, F10, F9	98%	11.25%	0.9423	0.98	0.9607

**Table 5: Detection Results on Unknown Samples**

	GoldDream	ADRD	jSMSHider	RogueLemon	DroidKungFu4	DroidKungFu5	Total
Using 9 Features	85%	85%	93%	50%	75%	100%	83.3%
Using 22 Features	90%	85%	93%	50%	83.3%	100%	87.25%

the minimal set which give fast detection as well as high detection rate.

#### 4.5 Limitations and Discussion

From the results we can see that 146 and 147 malicious samples are detected correctly out of 150 malicious samples by considering 9 features and all features respectively. The samples which are not detected correctly belong to malware families of DroidKungFu, Pjapps and Geinimi. One of the common thing in these malware families is the use of obfuscation techniques like encryption to communicate with their remote server. For instance Geinimi applies DES encryption algorithm to communicate with its remote server. Similarly

DroidKungFu sample applies AES encoding to encrypt its communication. Pjapps on other hand uses its own encoding scheme to encrypt address of its remote server and hide its communication. To evade detection attacker can use such obfuscation techniques so that malicious traffic looks similar to normal mobile traffic. So proposed approach fails in detecting samples which use encryption to communicate with its remote server. Out of total of 80 normal samples our approach detects 73 samples correctly and the remaining 7 result in false positives of online gaming applications (3 samples), online news (2 samples) and normal web browsing (2 samples). All these traffic samples generate features closely matching with that of malware. For example their packet

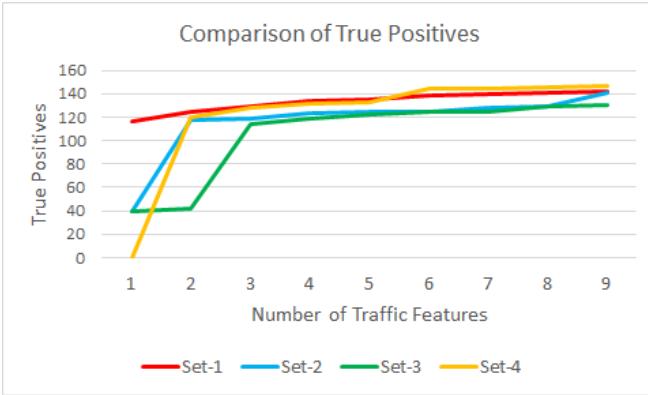


Figure 1: Comparison of True Positives

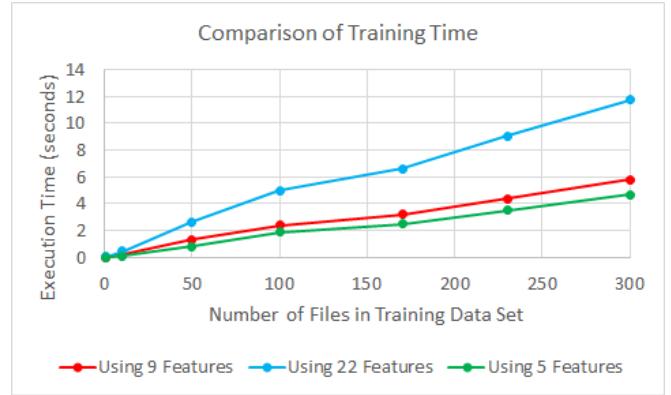


Figure 3: Comparison of Training Time

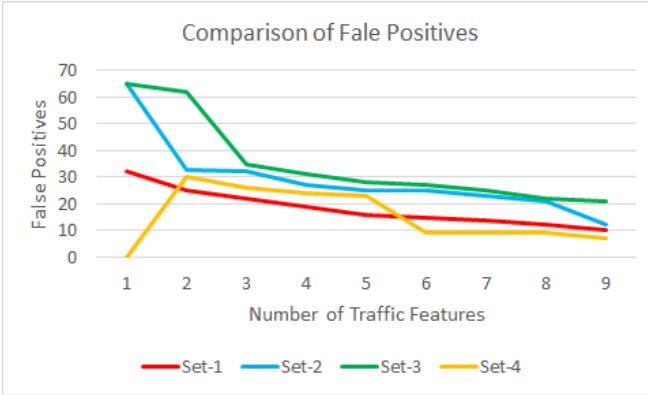


Figure 2: Comparison of False Positives

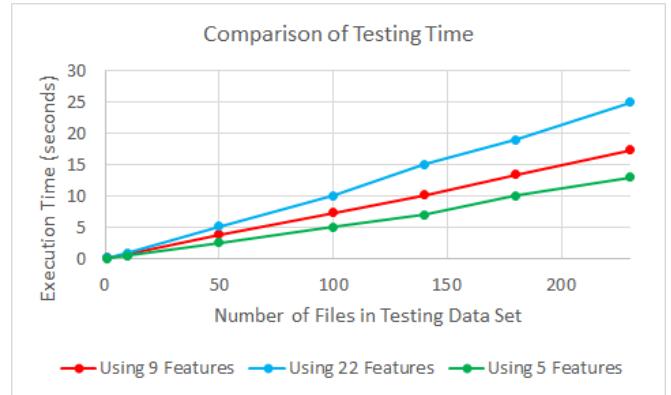


Figure 4: Comparison of Testing Time

size ( $F1$ ) is relatively low around 100 bytes; ratio of incoming to outgoing bytes ( $F2$ ) is comparatively less around 0.9; Maximum packet size ( $F16$ ) is also lower than other normal samples around 70 bytes etc. So such normal samples with low value of features like average packet size and amount of bytes may be classified as malicious samples.

Malware traffic for each sample was captured for duration of 3 days i.e. around 72 hours traffic for each malware. Continuous internet connectivity on mobile device is required to capture their network traffic which can be a challenge in normal life if we want to detect the presence of malicious apps on any smartphone.

Malware like AnserverBot and BaseBridge wait for system related events like reboot of phone to complete before they trigger their malicious payloads on the mobile device. Since we knew about their behavior, so all such events were completed while capturing their network traffic. Normal user in some cases may not be aware of such activities and hence may not complete that activity or does that activity after a long duration of time; so the malware although has infected the phone; will not produce any network traffic and hence it will evade detection for that time.

This approach is an off-device detection i.e. although traffic samples of apps are captured from smartphone but analyzed at a different system rather than on the smartphone. Doing on device detection consumes resources of the smartphone like memory, CPU, battery etc but provides real time detection. This work, however, does not provide real time

detection which can be a future work of this approach.

The experiments are conducted on malware samples that were detected around 2011. Since then there has been a significant increase in number of Android malware. So we will extend this work to recent malware samples as well.

## 5. CONCLUSIONS

In this paper we have prioritized traffic features out of total of 22 feature set based upon the rankings from Information Gain and Chi-Square Tests. A novel algorithm is presented to extract most important features for detection from the rankings obtained by information gain and Chi-square tests. The proposed feature selection algorithm extracts 9 important features out of total of 22, which give high detection accuracy. We have shown that prioritized features not only give best detection results but also save training and testing time. However, few samples using obfuscation techniques evade detection as they use encryption for communication with remote server. In our future work we will try to extend this work to recent Android malware samples as well as to other mobile platforms as well. We will also target to analyze encrypted network traffic of malware samples using techniques like deep packet analysis, because using obfuscation techniques has been common in recent malware samples to evade detection. We will also look further to provide a real time and lightweight traffic monitoring malware detection tool for mobile devices based upon analysis

of prioritized features.

## 6. ACKNOWLEDGMENTS

We would like to thank Yazin Zhou and Xuxian Jiang for providing us the malware data set through Android Malware Genome Project. We also acknowledge RailTel Telecom Center of Excellence (grant code RCI-763(3)-ECD) and IBM (grant code IBM-741-ECD) for providing funds to procure necessary equipment and other components.

## 7. REFERENCES

- [1] Statista. Number of smartphone users worldwide from 2014 to 2019 (in millions). <http://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/>, 2016. [Online; accessed 01-March-2016].
- [2] Smartphone OS Market Share, 2015 Q2. <http://www.idc.com/prodserv/mobile-os-market-share.jsp>, 2015. [Online; accessed 01-March-2016].
- [3] Y. Zhou and X. Jiang. Dissecting android malware: Characterization and evolution. In *2012 IEEE Symposium on Security and Privacy (SP)*, pages 95–109, May 2012.
- [4] A. Arora, S. Garg and S. K. Peddoju. Malware detection using network traffic analysis in android based mobile devices. In *8th International Conference on Next Generation Mobile Apps, Services and Technologies (NGMAST)*, pages 66–71, Sept 2014.
- [5] W. Enck, M. Ongtang and P. McDaniel. On lightweight mobile phone application certification. In *16th ACM conference on Computer and Communications Security*, pages 235–245, 2009.
- [6] S.H. Seo, A. Gupta, A. M. Sallam, E. Bertino, and K. Yim. Detecting mobile malware threats to homeland security through static analysis. *Journal of Network and Computer Applications*, 38:43 – 53, 2014.
- [7] V. Moonsamy, J. Rong, and S. Liu. Mining permission patterns for contrasting clean and malicious android applications. *Future Generation Computer Systems*, 36:122 – 132, 2014.
- [8] K. Xu, Y. Li and H.R. Deng. ICCDetector: ICC-Based Malware Detection on Android. *IEEE Transactions on Information Forensics and Security*, 11:1252–1264, 2016.
- [9] W. Wang, X. Wang, D. Feng, J. Liu, Z. Han and X Zhang. Exploring permission-induced risk in Android applications for malicious application detection. *IEEE Transactions on Information Forensics and Security*, 9:1869–1882, 2014.
- [10] E. William, G. Peter, C. Byunggon and C. Landon. TaintDroid: An information flow tracking system for realtime privacy monitoring on smartphones. *Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation*, 2011.
- [11] I. Burgera, U. Zurutuza and S. Nadjm-Tehrani. Crowdroid: Behavior-Based Malware Detection System for Android. *Proceedings of the 1st Workshop on Security and Privacy in Smartphones and Mobile Devices*, 2011.
- [12] Y. Lin, Y.C. Lai, C.H. Chen, and H.C. Tsai. Identifying android malicious repackaged applications by thread-grained system call sequences. *Computers and Security*, 39, Part B:340 – 350, 2013.
- [13] A. Shabtai, L. Tenenboim-Chekina, D. Mimran, L. Rokach, B. Shapira, and Y. Elovici. Mobile malware detection through analysis of deviations in application network behavior. *Computers and Security*, 43:1 – 18, 2014.
- [14] Z. Chen, H. Han, Q. Yan, B. Yang, L. Peng, L. Zhang, and J. Li. A first look at android malware traffic in first few minutes. In *2015 IEEE Trustcom/BigDataSE/ISPA*, pages 206–213, Aug 2015.
- [15] J. Li, L. Zhai, X. Zhang, and D. Quan. Research of android malware detection based on network traffic monitoring. In *2014 IEEE 9th Conference on Industrial Electronics and Applications (ICIEA)*, pages 1739–1744, June 2014.
- [16] A. Feizollah, N. B. Anuar, R. Salleh, and F. Amalina. Comparative study of k-means and mini batch k-means clustering algorithms in android malware detection using network traffic analysis. In *2014 International Symposium on Biometrics and Security Technologies (ISBAST)*, pages 193–197, Aug 2014.
- [17] M. Zaman, T. Siddiqui, M. R. Amin, and M. S. Hossain. Malware detection in android by network traffic analysis. In *2015 International Conference on Networking Systems and Security (NSysS)*, pages 1–5, Jan 2015.
- [18] G. Hu and D. Venugopal. A malware signature extraction and detection method applied to mobile networks. *IEEE International Performance, Computing, and Communications Conference*, pages 19–26, 2007.
- [19] R. Jin and B. Wang. Malware detection for mobile devices using software-defined networking. *8th GENI Research and Educational Experiment Workshop*, pages 81–88, 2013.

# Contemporary IT Security for Military Online Collaboration Platforms

Günter Fahrnberger

University of Hagen

North Rhine-Westphalia, Germany

guenter.fahrnberger@studium.fernuni-hagen.de

## ABSTRACT

Persistent armed conflicts with volatile intensities around Europe and resultant incidents directly on the *old* continent prompt the European governments and their general staffs to hold militiamen as reserve forces ready for emergency cases. These semi-professional soldiers need to regularly exercise their military skills. It is common practice that the leaderships of militia organizations connect via the Internet to military online collaboration platforms to prepare their field exercises there. Despite the involvement of the Internet in the transmission of sensitive military data, the IT security objectives authenticity, integrity, nonrepudiation, privacy, and resilience must be achieved at all costs. In the absence of an apposite IT security concept for such platforms in the literature, this publication amalgamates topical techniques to propose an access model that fulfills all five IT security targets. Beside the development stages *threat model*, *security policy*, and *security mechanism*, the disquisition proves the feasibility of the approach by manifesting airtight performance and security analyses.

## Keywords

Authentication, Authenticity, Availability, Biometry, Collaboration, Confidentiality, Cooperation, ECG, Electrocardiogram, Electrocardiography, Integrity, Military, Nonrepudiation, Privacy, Protection, Resilience, Secrecy, Security

## 1. INTRODUCTION

The long-standing trouble spots near Europe's borders (Ukraine, Syria, Iraq), the entailing flow of refugees, and the latent danger of terrorism evidence that the executed or still ongoing personnel and material disarmament of armed forces in European countries risks the successful coping of these challenges. Furthermore, in the recent decades many European governments have enacted the abolishment of conscription in their dominions. Despite this trend, some nations still cling to militia as a reserve of their professional

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

*ICDCN '17, January 04 - 07, 2017, Hyderabad, India*

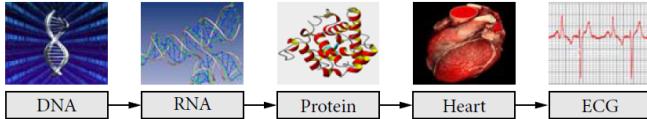
© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007754>

forces. Albeit militiamen mainly pursue their civilian professions and simultaneously hold themselves in readiness to be committed to new or running missions, they must attend regular trainings to maintain and improve their military skills. Such field exercises demand intensive planning from the command – the staff officers and their commandant in case of a battalion or of a larger military organization. This needful preparation lets them come together for a few inevitable meetings in military areas, but they process the vast bulk of preparative work distributed in their home or job environments. Due to the unavailability of secured military data networks outside of barracks, the officers may take advantage of nonmilitary online media for communication and data exchange as long as they obey all relevant provisions. The legal use cases are rare because classified and secret information must not traverse any civil transmission paths. It can happen that the soldiers inadvertently perpetrate the effectual privacy policy out of habit, convenience, or ignorance, e.g. if they convey sensitive documents via private e-mail accounts or online shares.

This shortcoming has evoked the emergence of military online collaboration platforms on which commanders of militia organizations and their staff can log in via Internet, interchange data, and communicate with each other. Such platforms are also utilized if various allied forces must temporarily collaborate in trainings or missions. It goes without saying that the IT (Information Technology) security of these platforms and their client devices must be prudently designed because of their leg to the Internet. Prudent design means the achievement of the IT security goals authenticity, integrity, nonrepudiation, privacy, and resilience. In the absence of such a concept in the literature, the disquisition at hand amalgamates topical techniques to suggest a model for the access to a cooperation system that fulfills all five IT security targets.

While privacy signifies that merely entitled officers can read confidential information which they gain from or place on the collaborative system, integrity connotes that unauthorized persons cannot modify them. Both aims are attained with platform-side MLS (MultiLevel Secure) systems [2] and state-of-the-art cryptosystems during data conveyance. Methodical logging of user activities through the platform guarantees nonrepudiation, viz. authorized users cannot deny their actions later. Resilience denotes the ability to provide high availability in spite of physical and DoS (Denial of Service) attacks. Geographical redundancy (multiple synchronized instances of the platform at geographically distributed locations) does the trick to preserve resilience. The seminal



**Figure 1: Inheritance model of ECG biometric from DNA biometric [42]**

merit of the treatise on hand is the profound elaboration of authentication (verification of pretended user identities) for the cooperation platform. Every officer must undergo continuous three-factor authentication during their sessions on the platform by presenting their correct login credentials (knowledge factor) and using their personalized ECG (ElectroCardioGram) sensor (possession factor) that taps and submits their ECG (inherence factor).

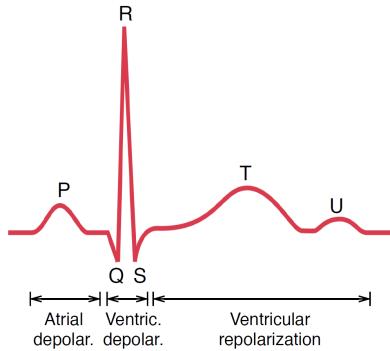
For that purpose, section 2 chronologically describes existing research results about biometric identification and authentication by dint of electrocardiography. In accordance with the recommended development stages *threat model*, *security policy*, and *security mechanism* [2], section 3 characterizes the threats to a collaboration platform, section 4 comes along with appropriate countermeasures as well as, particularly, the decisive reasons for authentication via ECG, and section 5 shows the implementation details of a proper prototype. Section 6 delivers insight into the efficiency of the suggested safeguards with the upshot of a conducted performance analysis. Ultimately, section 7 dedicates itself to a security analysis of the proposed precautions and fades out with worthwhile future work.

## 2. RELATED WORK

An ECG graphically records the electrical impulses of a creature's heart and represents the main tool to discover cardiac arrhythmia and other heart diseases. Figure 1 drafts the inference that ECG and other biometric entities eventually inherit their uniqueness from the individuality of DNA [42]. Apart from biometric purposes, a gaugeable ECG signal also implies a clear sign of life. Moreover, it can be even obtained from amputees who have lost some biometric marks (such as ears, fingers, hands, or voice) or who have congenitally lived without them.

The Dutch physician Willem Einthoven introduced the international labeling of the distinctive deflections of an ECG trace with different letters as depicted in figure 2 [17]. The character *P* earmarks the first wave of an ECG that initiates depolarization (rapid positive electrical cell charge) and, thereby, (mechanical) contraction of both atria. The so-called QRS complex denominates the subsequent spike (starting at the position symbolized by the character *Q*, peaking at the point labeled by the character *R*, and finishing at the spot indicated by the character *S*) that causes depolarization and, thus, contraction of both ventricles. The characters *T* and *U* tag the last two wavelike stimuli that induce repolarization (rapid negative electrical cell charge) and, hence, (mechanical) relaxation of both ventricles. The impulse for atrial repolarization respectively relaxation supposedly occurs during ventricular contraction and, thence, hides behind the more vigorous QRS complex.

Already in 1961, Stallmann and Pipberger developed a digital computer program for the automatic recognition of



**Figure 2: Heart activity during an ECG wave [39]**

electrocardiographic waves as the basis for a complete electrocardiographic analysis [40].

40 years later, Biel et al. were the prominent pioneers who indeed identified successfully their 20 test persons with the help of multivariate analysis of 360 extracted fiducial indexes gathered from the probands' 12-lead rest ECG recordings [8]. Fiducial properties (synonymously referred to as direct time domain features) incorporate intra- and inter-beat durations, amplitudes, curvatures, directions, and slopes distilled from a recorded ECG [42]. In contrast, non-fiducial attributes embrace frequency domain features (achieved via miscellaneous signal processing methods, like Fourier transformation, wavelet transformation, or discrete cosine transformation), statistical techniques, neural network and polynomial based approaches [42]. A suitable algorithm successfully identifies a testee if the features quantified during their enrollment and those ones measured during their examination resemble within defined thresholds. The experiments of Biel et al. also proved the possibility of identifying a human by fiducial attributes gauged from one lead and, accordingly, by means of only three body electrodes [8].

Shen et al. verified the proof of Biel et al. with ECGs of 20 individuals from the MIT/BIH database [18, 37]. They combined the 95% identification rate of template matching with the 80% identification rate of a DBNN (Decision Based Neural Network) to a 100% subject identification rate.

The introduction of AMON (a wearable multi-parameter medical monitoring and alert system) signified a step toward mobile ECG measurement [3, 28]. Its devisers especially engineered it for high risk patients rather than for biometric identification or authentication. The wrist worn medical monitoring computer can measure one ECG lead and supports an additional external sensor that allows to sample full 12-lead ECGs.

While all former suggestions just used ECGs of resting subjects for their investigations, Israel et al. successfully identified between 97% and 98% of their 29 probands irrespective of a low or high anxiety state [20].

Plataniotis et al. elaborated the first approach for ECG identification based on non-fiducial properties [33]. Their AC/DCT method attained a recognition rate of 92.86% by applying normalized Euclidean distance and 100% by exerting normalized Gaussian log likelihood for 14 healthy subjects from the PTB database [18].

Wang et al. employed the methodologies of Israel et al. [20] and Plataniotis et al. [33] with 13 testees from the PTB and MIT-BIH databases for the sake of comparison and achieved

perfect subject identification rate with both of them [45]. Molina et al. authored the first paper about authentication based on ECG rather than about identification [31]. They scanned one lead ECGs of ten individuals with two electrodes placed on both wrists and scored an EER (Equal Error Rate) of 2% with fiducial characteristics.

Chan et al. came up with a novel idea for sensing [9]. Each of their 50 volunteers held a pair of electrodes between the pads of their thumbs and index fingers. Non-fiducial distance measurement based upon wavelet coefficients applied to the collected ECGs led to classification accuracies of 89%, and as high as 95% when misclassifications were retested utilizing new data.

The proposal of Sriram et al. embodied two novelties [39]. First, it based upon hybrid properties, viz. fiducial and non-fiducial ones. Second, it considered the physical strain intensity of the 17 consulted volunteers by enriching up their biometric profile with their movement data from a triaxial accelerometer during workout sessions. The person identification accuracy laid between 82.35% and 88.23% dependent on the number of incorporated activity levels. The person authentication precision settled down between 94.12% and 100% depending on the amount of simulated impostors.

Two author teams extended conventional single time instant authentication to long term respectively continuous authentication. Coutinho et al. adopted their previous string matching approach based on non-fiducial characteristics and effectuated perfect verification rate with their 19 probands [10]. Labati et al. chose fiducial template matching and got out a minimum EER of 1.59%, a maximum EER of 8.55%, and an overall EER of 5.36% during their experiment with 185 ECG recordings [24].

Shen et al. exploited the palms as source for a one lead ECG [38]. Identification via template matching of 17 fiducial attributes succeeded for 95.3% of 168 probands.

Safie et al. bred PAB (Pulse Active Bit) feature extraction – a way that generates convenient hybrid properties for biometric systems which have limited storage capacity but deal with large numbers of subjects [35]. The characteristic vector for an individual consists of the intersection points between their ECG signal and a superimposing artificial triangular wave. The authors tested their suggestion with the ECGs of 113 humans from the PTB database and reached an EER of 31.17% [18]. They successively lowered this sub-optimal EER in their successional articles.

Da Silva et al. had the inventive conception for portable ECG signal acquisition based on electro-textile electrodes to avoid pre-gelled or medical grade electrodes [12]. Both Zelt and Electrolycra electro-textile materials exhibited accurate outcome when deployed as electrodes.

Mansfield-Devine reported on the ambitious US military's plans of combining biometric recognition systems with sensors capable of reading heartbeats through walls and even thick concrete with more than 95% confidence [30].

The recent proposals tended to optimize authentication and identification accurateness by conjoining electrocardiography, each with another biometric option. Zokae and Faez devised an amalgamation of palmprint and non-fiducial ECG variables to identify 47 patients from the MIT-BIH database and 50 subjects from the Holter database with detection rates between 94.7% and 100% [18, 46]. Derawi and Voitenko fused electrocardiography and gait for biometric authentication with EERs between 1.26% and 1.7% at best and EERs

between 6.5% and 7.3% in the most improved cases [14]. Manjunathswamy et al. accomplished multimodal biometric authentication by fusing fingerprint minutiae and fiducial ECG characteristics with an obtained exactitude rate of 97.5% [29]. Belgacem et al. presented the fusion of EMG (ElectroMyoGram) and non-fiducial ECG signals with an exactness of 99.98% for 60 attendant volunteers and 554 recordings from five Physionet databases [6, 18]. An EMG illustrates the electrical signal resulting from muscle contraction and becomes sampled by skin surface electrodes. A good overview of existent academic papers about recommendations for authentication and identification through electrocardiography and their subtleties can be won with the comparative surveys from Odinaka et al. [32] and Abo-Zahhad et al. [1].

Prior to the consideration of electrocardiography as an authentication factor, the next section explicates the entitative IT security threats against military cooperation platforms as pivotal cause for the implementation of a safe user verification process.

### 3. THREAT MODEL

Just as all IT systems, a military online cooperation platform can be threatened by nature and by humans. Natural threats (like earthquakes, fire, floods, and other disasters) rather precipitate the mere loss or damage of hardware resources and, therefore, jeopardize the service resilience. Anthropogenic menaces to hardware components encompass physical intrusion, physical sabotage, and weapon effects. Beyond that, lack of IT security knowledge and awareness (which platform users and administrators should have) as well as external and internal adversaries imperil the service on the software level and, for that reason, endanger the accomplishment of all five IT security objectives.

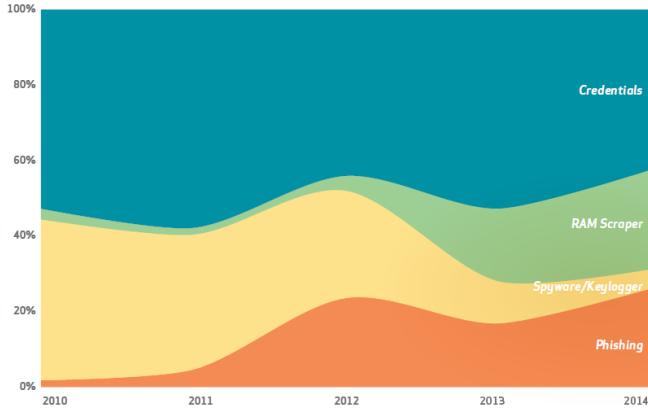
The question arises how skilled the attackers and the defenders of a military online collaboration platform might be and, on this account, which offenses can be expected respectively which counteractions do assaulters have to dread. The answer heavily depends on the value of a targeted system. Even though an offender would merely reveal, garble, or obliterate data of military trainings, these data shed light on the organizational chart and on the processes which ordinarily persist in real missions. It would hardly be an exaggeration to say that blackmail of military personnel by kidnapping their family members or disclosed tactics might be conceivable sore consequences. Hostile reconnaissance and intelligences go great lengths to harvest, replace, or eradicate such valuable information. It can be assumed that all of Barnard's aggressor types try their luck to break the military platform [4]. The lineup ranges from unskilled script kiddies who want to have fun and do not need to fear prosecution in case they are underage of criminal responsibility to highly skilled militants with university education serving in antagonistic forces. On that account, the military system must be envisaged and its lawful protagonists must be trained in a manner to withstand all imaginable raids.

Table 1 divulges the impact of diverse endangerments on the five established IT security objectives.

Nonphysical sabotage and all other physical threats peril the service availability. Imposed malware on client appliances or on the platform, ARP (Address Resolution Protocol) poisoning in the transmission network as well as platform-oriented (Distributed) DoS attacks through packet flooding or SQL

**Table 1: Threat Model**

Security Objective	Client Threats	Network Threats	Platform Threats
<b>Authenticity</b>	Impersonation	Session Hijacking	Compromise
<b>Integrity</b>	Impersonation	Session Hijacking	Compromise
<b>Nonrepudiation</b>	Impersonation	Session Hijacking	Compromise
<b>Privacy</b>	Eavesdropping Impersonation	Session Hijacking Sniffing	Compromise
<b>Resilience</b>	Impersonation Nonphysical Sabotage Physical Threats	Nonphysical Sabotage Physical Threats Session Hijacking	Compromise Nonphysical Sabotage Physical Threats



**Figure 3: Significant threat actions over time by percent [44]**

(Structured Query Language) injection typify examples for nonphysical sabotage. Besides natural hazards, physical threats also enfold (anthropogenic) physical sabotage, like theft and robbery of client devices as well as weapon effects against network and platform units.

Successful eavesdropping and sniffing of critical information (in computers, routers, and at air or tethered interfaces) sorely violate at least the required data confidentiality. Much worse, if login credentials thereby fall into the wrong hands, impersonation and session hijacking become very doable and prevent the attainment of any of the five IT security requirements. Just as well, cryptanalysis (inclusive exhaustive brute force offensives), Phishing (Password harvesting fishing), social engineering, or spoofing can achieve impersonating victims. Man-in-the-middle-attacks represent the most famous sort to intercept and hijack active sessions on the transmission path.

The worst case accrues if culprits accomplish compromising the military online platform itself, for instance through backdoor exploitation, malware, or software manipulation. In addition to remote appropriation of the contained data, they normally proceed to escalate their privileges for continuing hits and conclude to cover their tracks by destructing logfiles for example.

Figure 3 demonstrates the threat histogram of Verizon's most recent data breach investigations report [44]. It exposes purloined login credentials as the most significant attack vector nowadays. This emphasizes the importance of prudential multi-factor authentication specified in the beginning of the following subsection about security policy.

## 4. SECURITY POLICY

The antecedent section concerned instruments to wage offensive information warfare against a military online co-operation system. This kind of warfare refers to the use of ICT (Information and Communication Technology) to take advantage over an opponent's ICT assets. This section treats the opposing means of defensive information warfare to correspondingly protect a military collaboration platform against the offensive counterpart.

It is clear as daylight that the rigorous fulfillment of all five IT security goals can entail less functionality and usability. Nonetheless, the handling must be yet simple enough to perpetuate wide user acceptance. Each of the five below-mentioned (sub)policies conforms to this simplicity rule. Otherwise, the users would bail out and seek more comfortable alternatives.

### 4.1 Authentication Policy

Usually, armed forces rely on security through obscurity rather than on security by design (open security based on Kerckhoffs' principle [22, 23] and Shannon's maxim [36]) for their IT infrastructure, i.e. they commonly conceal the MLS system(s) and the hybrid cryptographic scheme(s) which they have implemented to attain privacy and integrity. For this reason, scientific databases and other online sources cannot showcase technical details of currently running MLS systems and the applied cipher for their defense.

Nevertheless, a modern password policy for login credentials as the first of three authentication factors can be adopted from the civil domain. It must merely permit the usage of complex passwords which adhere to a minimal length and do not inclose easily guessable strings, such as dictionary expressions, calendar dates, telephone numbers, or user names. Passphrases with minimum quotas of lowercase characters, uppercase characters, numerical digits, and special characters have become accepted as best practice. Above all, the directive must technically enforce a maximal watchword lifetime and deter a user from setting passwords which they already had in use recently.

Beside password protection, biometry as the second of three authentication factors additionally ensures the authenticity of users who operate on the collaboration system. Biometric features comprise quantifiable physiological and behavioral characteristics which can be exerted for the identification and authentication of humans. While biometric identification determines a user identity on the basis of their given traits, biometric authentication decides whether the provided attributes originate from a certain individual.

Jain et al. comprehensively itemized and classified almost all available biometric variables with respect to seven qualifi-

**Table 2: Comparison between physiological biometric traits [1, 21]**

Biometric Identifier	Universality	Distinctiveness	Permanence	Collectability	Performance	Acceptability	Circumvention
DNA	H	H	H	L	H	L	L
Ear	M	M	H	M	M	H	M
Electrocardiography	H	H	H	M	H	M	L
Face	H	L	M	H	L	H	H
Facial Thermogram	H	H	L	H	M	H	L
Fingerprint	M	H	H	M	H	M	M
Hand Geometry	M	M	M	H	M	M	M
Hand Vein	M	M	M	M	M	M	L
Heart Sounds	H	M	L	L	L	M	L
Iris	H	H	H	M	H	L	L
Odor	H	H	H	L	L	M	L
Palm Print	M	H	H	M	H	M	M
Retina	H	H	M	L	H	L	L

**Table 3: Comparison between behavioral biometric traits [1, 21]**

Biometric Identifier	Universality	Distinctiveness	Permanence	Collectability	Performance	Acceptability	Circumvention
Gait	M	L	L	H	L	H	M
Keystroke	L	L	L	M	L	M	M
Signature	L	L	L	H	L	H	H
Voice	M	L	M	L	L	M	H

ties [21]. Abo-Zahhad et al. added ECG and heart sounds as biometric resources and subdivided the conglomeration of all elements in physiological (see table 2) and behavioral ones (see table 3) [1]. Any military entity, which can contain top secret content, must comply with the highest possible security standards. As a result, all biometric identifiers drop out, whose feasibility of circumvention counts as low or medium in the itemizations of Jain et al. and Abo-Zahhad et al. This disqualifies all behavioral traits (all items of table 3) and five options of table 2 (recognition via ear, face, fingerprint, hand geometry, and palm print). The maximization of permanence as the second most important factor eliminates biometric detection via facial thermogram, hand vein, retina, and heart sounds. DNA (DeoxyriboNucleic Acid) and odor become excluded due to collectability as the next selection criterion. The choice between the last two residual candidates *iris* and *ECG* falls on the latter because it can be comfortably sensed even from a user who moves in their office space in contrary to iris scans.

Due to the release of enhanced ECG authentication schemes each month, this authentication policy does not focus on a particular generation of user templates with fiducial or non-fiducial variables which become compared during user accreditation. Rather, it recommends one of the models in the overviews of Odinaka et al. [32] or Abo-Zahhad et al. [1] with high authentication rate respectively low EER.

The intended ECG measuring device as the third of three authentication factors must facilitate typical movements of a user in an office to foster their acceptability rather than handicap them with finger sensors or stick them on their

chair, or even worse, on their keyboard with stationary sensing pads [9, 11, 26, 27]. The most auspicious technique seems to be the employment of an AMON [3, 28] aforementioned in section 2 about related work. To overcome poor results due to high levels of measurement noise, a refined AMON version needs to realize upgraded bandwidth filtering for better noise reduction [3]. The unimproved AMON operates with the cutoff frequencies 0.05 and 100 Hz. The switched on bracelet digitalizes all metered voltages of the ECG signal and transmits the filtered values via Bluetooth protocol to the operator terminal.

The parallel establishment of these three authentication factors can unambiguously approve registered users and reject all other access attempts. That is why it evades implementing laborious certificate management with a PKI (Public Key Infrastructure).

## 4.2 Integrity Policy

Nobody can entirely preclude a villain from truncating crucial military data (such as ECG leads, credentials, and all other user data) during their transport from the client through the Internet to the military online cooperation platform respectively vice versa, but even slightest malicious alterations must arouse automated remedial measures or the attention of military employees.

In the case of TCP (Transmission Control Protocol) segments which carry plaintext, a hacker can airily manipulate the payload at will with good chances of remaining undiscovered as long as they properly adjust the correspondent checksums. False TCP checksums would spark off retrans-

mission of the affected segments.

If a contemporary hybrid cryptosystem in ECB (Electronic CodeBook) mode [19] scrambles the payload, an attacker can also arbitrarily alter some bits including adequate checksum adaptation without causing TCP segment transmissions. Such an offense would most likely provoke an application protocol error. Insistent ciphertext mutilations in consecutive TCP segments of the same session deprive the pertained user of their platform access. More seriously, an adversary who has unveiled the position and meaning of one or more ciphertext blocks in the stream of TCP segments (by way of example through collision offenses which have capitalized repeated ciphertext blocks) can purposefully substitute it/them for maleficent replacement with cut-and-splice-offensives [2].

For this purpose, the exercised encryption scheme must build independent ciphertext blocks by tying the first block with a random seed, by coalescing all other blocks with their predecessors, and by padding the very last block with arbitrary bits to block length if needed. Several modes of operation (e.g. CBC (Cipher Block Chaining), CFB (Cipher Feed-Back), or OFB (Output FeedBack)) support such secure ciphering [19].

The jeopardy of integrity violations in clients must be mitigated through locally installed firewalls and malware scanners with perpetual up-to-date threat signature repositories.

### 4.3 Nonrepudiation Policy

Nonrepudiation warrants that platform users cannot deny their performed activities later. Relying upon the enforcement of the previously characterized authentication policy, a conformable nonrepudiation policy can be kept short and simple. It only needs to coerce the platform noting down all received user commands in a logfile or in a database.

### 4.4 Privacy Policy

OTP (One Time Pad) would offer perfect secrecy for transported military information in terms of Shannon's communication theory [36] if an RRNG (Real Random Number Generator) creates a key stream with at least the same length as the input plaintext, but the requisite transfer of the key stream via a safe second channel to a receiver makes it unfeasible.

Robust instead of perfect privacy becomes attainable by merely adhering to one paradigm: disguise the largest possible bunch of plaintext together with compositions of substitution and transposition functions [5].

On account of this, the utilization of an approved-as-secure cryptographic suite with a fast symmetrical block enciphering scheme for plaintext scrambling and a generally slower asymmetric cryptosystem for the negotiation of symmetric session keys appears to be the most promising solution for interchanged data between a user terminal and the server platform. A viable combination would be if DH (Diffie Hellman) [16] or ECDH (Elliptic Curve Diffie Hellman) [41] produce session keys for AES (Advanced Encryption Standard) [13]. In compliance with the integrity policy, the plaintext en- and decipherments must unconditionally base on a mode of operation without vulnerability to ciphertext repetitions.

The concealment of a transmitted ECG lead between an AMON and its paired user device must also be assured notwithstanding its wireless signaling via Bluetooth that is

prone to bugging. On account of that, among initial credentials, the responsible authority hands a personalized AMON with a programmed pre-shared symmetrical key (that cannot be externally readout anymore) out to each legitimated platform user. The AMON applies the key to encrypt the digitalized amplitudes of the ECG lead with a symmetric ciphering scheme before it conveys the ciphered amplitudes via the client terminal to the platform. For the decryption of the ECG fragments, the platform employs the pre-shared key that it finds in its user repository for a purported user identity. An asymmetrical key stipulation was not planned to avert time-consuming cryptographic operations in the CPU (Central Processing Unit) of the AMON. To definitely obviate repetitive ciphertexts and resultant tempting collision assaults, a prepended timestamp acts as nonce (number used once) and makes each plaintext (and consequently each ciphertext) ECG fraction unique.

Likewise above-mentioned in the integrity policy, privacy breaches directly in a client must be thwarted with firewalls and updated malware scanners operating on it.

### 4.5 Resilience Policy

Platform users can freely decide with which workstation they want to connect to their online collaboration platform as long as it proffers the requested safety precautions (local firewall and malware scanner), Internet and Bluetooth functionality. In the event of its breakdown, they can simply revert to another equally equipped appliance.

In case the AMON of a user malfunctions, they cannot make use of a fellow's AMON in lieu of their failed one because it would encipher their ECG with an incorrect pre-shared key, and the platform fails to authenticate them. Only the programming and issuance of an intact spare part remedies this deficit.

This resilience policy does not need to specify dedicated safeguarding for the ISPs (Internet Service Providers) which the military staff utilize. On the one hand, a plurality of ISPs in industrial nations proffer sufficient redundancy to absorb sporadic outages. On the other hand, rational ISPs have realized their own resilience policy to keep their services alive at all events.

The online platform must run on a geographically distributed cluster with one active node and the other ones in standby condition to preserve data consistency in face of concurrently working users. All replicas must be located in computing centers in military zones with strict physical ingress control, and their IP (Internet Protocol) traffic must be screened by firewalls with appropriately configured ACLs (Access Control Lists).

The following section brings all five subpolicies of the just discussed security policy to fruition.

## 5. SECURITY MECHANISM

Figure 4 depicts the architectural model of a prototype with a sole client, which realizes the security mechanism derived from the security policy. Before a user can sign in to the platform as per subsection 5.3 and execute their tasks there under permanent ECG verification according to subsection 5.4, they must go through the enrollment phase at least once and get ECG transmission between their AMON and their client equipment started as described below in the subsections 5.1 and 5.2.

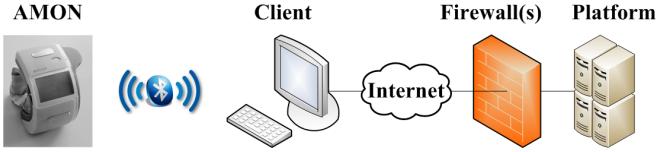


Figure 4: Architectural Model

## 5.1 Enrollment Phase

First and foremost, the platform administration squad adds a new user account that includes a randomized triple of login name, temporary password, and pre-shared key. Afterwards, the team programs an unused AMON with this key and hands the equipment together with the login credentials out to the account owner. The registration workflow ends with the recording of their reference ECG lead by starting up their personal AMON. After noise suppression through the bandpass filter in the AMON, the platform extracts all necessary fiducial and/or non-fiducial variables and attaches them to the user profile.

## 5.2 ECG Transmission

A wearer of AMON prepares the logon by turning it and their client appliance on and pairing them via Bluetooth. Once the coupling has succeeded, the AMON immediately begins to continuously sense, scramble, and transmit the ECG of its possessor to the client hardware. The on-board ADC (Analog Digital Converter) of the revised AMON converts picked up voltages to 16 bits long binary numbers whereas the original AMON produces only eleven bits long ones. The subtler resolution enables higher authentication success. The AMON senses one voltage every two milliseconds by reason of its sampling frequency of 500 Hz. It amasses all 500 measurements of each second in a new binary array and salts it by prepending the current timestamp as a 32 bits long unsigned integer nonce, viz. each array grows to a size of  $500 * 16 + 32 = 8032$  bits. The Unix time (number of elapsed seconds since 01.01.1970 00:00:00 UTC (Universal Time Coordinated)) qualifies for such a nonce because it just restarts every  $2^{32}$  seconds, i.e. approximately every 136 years, and that undoubtedly exceeds human life expectancy. It means that the first 32 bits of a 8032 bits long binary plaintext array do not recur in roughly 136 years. Afterward, the AMON uses its encapsulated pre-shared key to perform the encipherment function of AES for each binary plaintext array, viz. it pads each array to a length of 8064 bits with random bits and outputs  $63 (= \frac{8064}{128})$  cohesive 128 bits long ciphertext blocks. A safe mode of operation propagates the singularity of the nonce in the first block to all successive ones.

## 5.3 Logon Phase

At the outset of a logon procedure, the ECG transmission must be in progress. If fulfilled, the user opens the client application on their workstation in order to let it establish a secure bidirectional tunnel to the online cooperation platform with a trustworthy hybrid cryptoprotocol of the TLS (Transport Layer Security) suite [15]. Then they present their login name and their password when prompted. In case they enter the temporary codeword that was randomly assigned to them during the creation of their account, the platform coerces them to key in a novel one that complies

with the mentioned password policy in subsection 4.1.

## 5.4 Continuous ECG Verification

As soon as the platform acknowledges the receipt of valid unexpired credentials, the client application forwards all consigned ciphertext arrays from the AMON without any modification through the TLS tunnel to the platform nearly in realtime. This overhead of 8064 bits per second is more or less negligible with the today's data transfer rates. The server application on the platform decrypts the arrays with the pre-shared key that belongs to the profile of the confirmed user identity, evaluates all interesting fiducial and/or non-fiducial features out of the plaintext ECG signal, and compares them with the samples drawn during the enrollment phase. Once the discrepancy between the reference and the live attributes exceeds the set thresholds, the platform forcefully logs out the affected user and offers them a chance for relogin. The thresholds must be adjusted in a fashion that stress-induced anomalies in an ECG do not give rise to needless logoffs on the one hand, but they must be tight enough to refuse authentication with bogus heartbeats. It might be worth to consider forceful logouts concerning ECGs which apparently indicate abnormal pulse rates or huge stress symptoms because officers will not log on for (team)work during sport or extreme stress phases.

The penultimate section exhibits harvested results from a simulation with a prototype that implements the circumstantiated security mechanism.

## 6. PERFORMANCE ANALYSIS

The experiment-based performance analysis targets to time the latency period statistics on one second long ECGs between the AMON and the collaborative platform for various data bearer technologies. Thereto, the experimental setup instantiates the constituent parts in figure 4 in the following way.

- **AMON:** The AMON becomes emulated by a Java 1.8 process on a laptop computer (containing an Intel Core i5-3320M CPU with 4 \* 2.6 GHz and 8 GB RAM) that sequentially reads all locally saved ECG files of the 90 test persons from the ECG-ID database [18] and encrypts them with AES-128/CBC/PKCS5Padding based on a separate pre-shared key for each proband to one second long binary ciphertext arrays which it hands over to its Bluetooth stack. Each source record file envelopes a 20 seconds long ECG at a sampling rate of 500 Hz and a resolution of twelve bits, i.e. it encloses 10,000 contiguous sample intervals. Previous to the common encryption of 500 samples per second together with their timestamp as nonce, the Java process imitates the AMON by turning each twelve bits long value into a 16 bits long one. Thanks to the initial nonce, the initialization vector for CBC can be zeroed in the sending AMON process and in the receiving platform process without impairing the robustness of CBC. In the end, the AMON process draws and displays descriptive statistics on the runtime of one second long ECG fragments. Each timing for these statistics starts with the import of the first of 500 voltages from a local ECG file and stops with the acknowledgment of decipherment from the platform.
- **Bluetooth:** Like AMON, the Bluetooth link relies

on a software emulation with BlueCove JSR-82 rather than on genuine hardware [43]. The BlueCove emulator bases on Java code and utilizes RMI (Remote Method Invocation) for interprocess communication. Unless bound, the virtual AMON tries to resolve the interruption with periodic bind requests toward the client.

- **Client:** The computer simulating the AMON also runs the client in another Java process that listens to incoming bind inquiries from the AMON via Bluetooth and finalizes the mating cycle. Thereafter, it negotiates a common 128 bits long secret key with the platform via the Diffie Hellman key agreement method [34] and establishes a virtual secure tunnel toward the platform with AES-128/CBC/PKCS5Padding based on the negotiated secret key, i.e. it enciphers all tunneled data. Subsequently, the client process tunnels the user credentials via RMI to the platform and waits for the validation before it continues to tunnel ECG and other data as well via RMI. A locally installed and reasonably configured SELinux (Security-Enhanced Linux) on the laptop assures safety of the client [25].
- **Internet:** The mobile data network of T-Mobile Austria serves as ISP to bear enciphered information between the client and the platform. The experiment ascertains the mean throughput time statistics between client and platform for each of the three contemporaneously offered radio technologies EDGE (Enhanced Data rates for GSM (Global System for Mobile communications) Evolution), HSPA (High Speed Packet Access), and LTE (Long Term Evolution).
- **Firewall(s):** A locally installed and adequately configured SELinux (Security-Enhanced Linux) on the platform computer simulates the military firewall(s) between the Internet and the DMZ (DeMilitarized Zone) with the platform [25].
- **Platform:** A Java process on a publicly reachable virtual cloud computer with a single Intel Pentium 4 CPU 2.80 GHz and 1 GB RAM plays the role of a military online collaboration system. It saves the profile of each of the 90 testees in a local MariaDB table. The profile of each user incorporates their login name, their password (hashed with SHA3-512 [7]), the pre-shared key of their AMON, and their ECG template. The platform process deciphers all inbound data from the tunnel with the stipulated secret key and processes them fittingly. In the event it receives credentials, it hashes the password and likens the login name and the hashed password to the profile table. As soon as a scrambled array with an ECG second arrives, the platform process descrambles it, distills the essential properties, and (re)validates or falsifies the user genuineness. Further data do not incur during this experiment. Under non-experimental circumstances, the platform application would handle them.

Table 4 contains the descriptive runtime statistics for 12,400 processed ECG seconds per radio technology in consideration of the above stated experimental orchestration. While the speed of HSPA merely lags approximately ten percent behind that one of LTE, EDGE performs roughly six times

**Table 4: Descriptive Runtime Statistics**

Radio Technology	EDGE	HSPA	LTE
Minimum Runtime	273.86 ms	49.31 ms	43.90 ms
Medium Runtime	345.19 ms	66.07 ms	58.18 ms
Mean Runtime	356.77 ms	66.79 ms	58.68 ms
Maximum Runtime	1336.99 ms	450.12 ms	417.63 ms
Standard Deviation	177.59 ms	11.36 ms	9.50 ms

slower than LTE. Initially, the platform does not have information about the data bearer, the radio coverage, and the QoS (Quality of Service) of a connection to a user. On these grounds, the platform must tolerate ECG fragments delayed up to a few seconds before it terminates a user session. As soon as the platform detects that the average runtime levels off, it can determine a more restrictive delay threshold.

The endmost section reflects on the conformance of the prototypical security mechanism with the demanded IT security aims.

## 7. CONCLUSION

The most striking evaluation of the accomplished IT security requirements in the military cooperation system seems to be the derivation of implications in the case of compromised system constituents.

A malefactor could blackmail or compel an officer to log in to the platform and act on the malefactor’s behalf. Two safety procedures can balk that. Firstly, within the meaning of a panic button, the victim can deliberately take an inconspicuous predefined action in the user interface of the platform to covertly alarm the military surveillance on duty. Secondly, the officer’s stress influences their ECG in such an extent so that the platform does not any longer verify their authenticity and forcibly logs them off.

An assailant might succeed in catching a user’s ECG and feeding the platform with it. That would miscarry even with the right nonce because they do not know the pre-shared key to correctly scramble the ECG.

If the assaulter also gains physical control over the appendant AMON and lets it sense the stolen ECG records, then they are still unaware of the third authentication factor – the login credentials.

On the contrary, succeeding malware on the client can uncover the login credentials, but cannot imitate the ciphered ECG fragments.

Only the composition of thieved ECG recordings, of the physically controlled associated AMON, and of the desried appendant plaintext login credentials empowers an evildoer to counterfeit a user.

Further, the air interface with the Bluetooth datagrams can appeal miscreants to launch collision attacks. As soon as a collision attack on the encrypted output of an AMON becomes realistic, the signal-generating heart has already stopped beating because a nonce in front of every plaintext binary array with ECG voltages impedes recurring ciphertexts within nearly 136 years.

The routes in the Internet as well beckon to wiretap and unscramble client-platform-communication, but two obstacles make such an intrusion to a tough nut to crack. The TLS tunnel as the first hindrance protects all data units of a session. Over and above, a wrongdoer would also need to break the encipherment of the ECG fragments as the second barrier.

The platform with the stored multiplicity of military sensitive information poses the most luring and also the most difficult point of attack. At least one firewall between the Internet gateway and the platform confines illegal remote access attempts, and its positioning on guarded military area cares for ample physical protection.

Owing to the fact that the compromise of any sole component of figure 4 does not hamper to reach the IT security goals authenticity, integrity, nonrepudiation, privacy, and resilience, the writer of this scholarly piece remains of the conviction that it really lends itself to a practicable utility for military and other scopes which require high security through three-factor authentication.

However, a continuation of this scientific paper is strongly desirable that opts for a specific ECG authentication scheme and elaborates a holistic approach. The implementation with actual AMON devices would ameliorate evaluating the feasibility more realistically, as their maturity could be a limiting factor for adopting such an approach. It would be also interesting to assess how effectively the mechanism works under noisy conditions or when the heart rate of the user significantly changes (for example due to physical activity).

## Acknowledgments

Many thanks to Bettina Baumgartner from the University of Vienna for proofreading this paper!

## 8. REFERENCES

- [1] M. Abo-Zahhad, S. Ahmed, and S. Abbas. Biometric authentication based on PCG and ECG signals: present status and future directions. *Signal, Image and Video Processing*, 8(4):739–751, may 2014.
- [2] R. J. Anderson. *Security engineering - a guide to building dependable distributed systems* (2. ed.). Wiley, jan 2008.
- [3] U. Anliker, J. A. Ward, P. Lukowicz, G. Tröster, F. Dolveck, M. Baer, F. Keita, E. B. Schenker, F. Catarsi, L. Coluccini, A. Belardinelli, D. Shklarski, M. Alon, E. Hirt, R. Schmid, and M. Vuskovic. AMON: a wearable multiparameter medical monitoring and alert system. *Information Technology in Biomedicine, IEEE Transactions on*, 8(4):415–427, dec 2004.
- [4] R. L. Barnard. *Intrusion Detection Systems*. Gulf Professional Publishing, feb 1998.
- [5] F. L. Bauer. *Decrypted Secrets: Methods and Maxims of Cryptology*. Springer Publishing Company, Incorporated, 4th edition, dec 2010.
- [6] N. Belgacem, R. Fournier, A. Nait-Ali, and F. Berekci-Reguig. A novel biometric authentication approach using ECG and EMG signals. *Journal of Medical Engineering & Technology*, 39(4):226–238, apr 2015.
- [7] G. Bertoni, J. Daemen, M. Peeters, and G. van Assche. The making of KECCAK. *Cryptologia*, 38(1):26–60, jan 2014.
- [8] L. Biel, O. Pettersson, L. Philipson, and P. Wide. ECG analysis: a new approach in human identification. *Instrumentation and Measurement, IEEE Transactions on*, 50(3):808–812, jun 2001.
- [9] A. D. C. Chan, M. M. Hamdy, A. Badre, and V. Badee. Wavelet distance measure for person identification using electrocardiograms. *Instrumentation and Measurement, IEEE Transactions on*, 57(2):248–253, feb 2008.
- [10] D. P. Coutinho, A. L. N. Fred, and M. A. T. Figueiredo. ECG-based continuous authentication system using adaptive string matching. In *Proceedings of the International Conference on Bio-inspired Systems and Signal Processing*, volume 1, pages 354–359, jan 2011.
- [11] H. P. da Silva, A. L. N. Fred, A. Lourenço, and A. K. Jain. Finger ECG signal for user authentication: Usability and performance. In *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on*, pages 1–8, sep 2013.
- [12] H. P. da Silva, A. Lourenço, R. Lourenço, P. Leite, D. Coutinho, and A. L. N. Fred. Study and evaluation of a single differential sensor design based on electro-textile electrodes for ECG biometrics applications. In *Sensors, 2011 IEEE*, pages 1764–1767, oct 2011.
- [13] J. Daemen and V. Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer, nov 2001.
- [14] M. Derawi and I. Voitenko. Fusion of gait and ECG for biometric user authentication. In *Biometrics Special Interest Group (BIOSIG), 2014 International Conference of the*, pages 1–4, sep 2014.
- [15] T. Dierks and E. Rescorla. The transport layer security (TLS) protocol version 1.2. RFC 5246 (Proposed Standard), aug 2008.
- [16] W. Diffie and M. E. Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, nov 1976.
- [17] W. Einthoven. Über die deutung des elektrokardiogramms. *Pflüger's Archiv für die gesamte Physiologie des Menschen und der Tiere*, 149(1-3):65–86, nov 1912.
- [18] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):215–220, jun 2000.
- [19] ISO/IEC 10116:2006. Information technology – security techniques – modes of operation for an n-bit block cipher. International Organization for Standardization, feb 2006.
- [20] S. A. Israel, J. M. Irvine, A. Cheng, M. D. Wiederhold, and B. K. Wiederhold. ECG to identify individuals. *Pattern Recognition*, 38(1):133–142, jan 2005.
- [21] A. K. Jain, A. Ross, and S. Prabhakar. An introduction to biometric recognition. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(1):4–20, jan 2004.

- [22] A. Kerckhoffs. La cryptographie militaire. 9:5–38, jan 1883.
- [23] A. Kerckhoffs. La cryptographie militaire. 9:161–191, feb 1883.
- [24] R. D. Labati, R. Sassi, and F. Scotti. ECG biometric recognition: Permanence analysis of QRS signals for 24 hours continuous authentication. In *Information Forensics and Security (WIFS), 2013 IEEE International Workshop on*, pages 31–36, nov 2013.
- [25] P. Loscocco and S. Smalley. Integrating flexible support for security policies into the linux operating system. In *Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference*, pages 29–42, Berkeley, CA, USA, jun 2001. USENIX Association.
- [26] A. Lourenço, H. P. da Silva, and A. L. N. Fred. Unveiling the biometric potential of finger-based ECG signals. *Intell. Neuroscience*, 2011:1–8, jan 2011.
- [27] A. Lourenço, H. P. da Silva, D. P. Santos, and A. L. N. Fred. Towards a finger based ECG biometric system. In *Proceedings of the International Conference on Bio-inspired Systems and Signal Processing*, volume 1, pages 348–353, jan 2011.
- [28] P. Lukowicz, U. Anliker, J. Ward, G. Tröster, E. Hirt, and C. Neufelt. AMON: a wearable medical computer for high risk patients. In *Wearable Computers, 2002. (ISWC 2002). Proceedings. Sixth International Symposium on*, pages 133–134, oct 2002.
- [29] B. E. Manjunathswamy, A. M. Abhishek, J. Thriveni, K. R. Venugopal, and L. M. Patnaik. Multimodal biometric authentication using ECG and fingerprint. *International Journal of Computer Applications*, 111(13):33–39, feb 2015.
- [30] S. Mansfield-Devine. Biometrics at war: the US military's need for identification and authentication. *Biometric Technology Today*, 2012(5):5–8, may 2012.
- [31] G. G. Molina, F. Bruekers, C. Presura, M. Damstra, and M. van der Veen. Morphological synthesis of ECG signals for person authentication. In *Signal Processing Conference, 2007 15th European*, pages 738–742, sep 2007.
- [32] I. Odinaka, P.-H. Lai, A. D. Kaplan, J. A. O'Sullivan, E. J. Sirevaag, and J. W. Rohrbaugh. ECG biometric recognition: A comparative analysis. *Information Forensics and Security, IEEE Transactions on*, 7(6):1812–1824, Dec 2012.
- [33] K. N. Plataniotis, D. Hatzinakos, and J. K. M. Lee. ECG biometric recognition without fiducial detection. In *Biometric Consortium Conference, 2006 Biometrics Symposium: Special Session on Research at the*, pages 1–6, sep 2006.
- [34] E. Rescorla. Diffie-hellman key agreement method. RFC 2631 (Proposed Standard), jun 1999.
- [35] S. I. Safie, J. J. Soraghan, and L. Petropoulakis. Pulse active bit (PAB) feature extractor for ECG biometric authentication. In *Systems, Signals and Image Processing (IWSSIP), 2011 18th International Conference on*, pages 1–4, jun 2011.
- [36] C. E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, oct 1949.
- [37] T.-W. Shen, W. J. Tompkins, and Y. H. Hu. One-lead ECG for identity verification. In *Engineering in Medicine and Biology, 2002. 24th Annual Conference and the Annual Fall Meeting of the Biomedical Engineering Society EMBS/BMES Conference, 2002. Proceedings of the Second Joint*, volume 1, pages 62–63, oct 2002.
- [38] T.-W. Shen, W. J. Tompkins, and Y. H. Hu. Implementation of a one-lead ECG human identification system on a normal population. *Journal of Engineering and Computer Innovations*, pages 12–21, jan 2011.
- [39] J. C. Sriram, M. Shin, T. Choudhury, and D. Kotz. Activity-aware ECG-based patient authentication for remote health monitoring. In *Proceedings of the 2009 International Conference on Multimodal Interfaces, ICMI-MLMI '09*, pages 297–304, New York, NY, USA, nov 2009.
- [40] F. W. Stallmann and H. V. Pibberger. Automatic recognition of electrocardiographic waves by digital computer. *Circulation Research*, 9(6):1138–1143, nov 1961.
- [41] D. Stebila and J. Green. Elliptic curve algorithm integration in the secure shell transport layer. RFC 5656 (Proposed Standard), dec 2009.
- [42] F. Sufi, I. Khalil, and J. Hu. ECG-based authentication. In P. Stavroulakis and M. Stamp, editors, *Handbook of Information and Communication Security*, pages 309–331. Springer Berlin Heidelberg, feb 2010.
- [43] B. Team. Bluecove java library for bluetooth, 2008.
- [44] Verizon Enterprise Solutions. 2015 data breach investigations report, may 2015.
- [45] Y. Wang, F. Agrafioti, D. Hatzinakos, and K. N. Plataniotis. Analysis of human electrocardiogram for biometric recognition. *EURASIP J. Adv. Signal Process*, 2008, jan 2008.
- [46] S. Zokaei and K. Faez. Human identification based on electrocardiogram and palmprint. *International Journal of Electrical and Computer Engineering (IJECE)*, 2(2):261–266, apr 2012.

# Coalitional Game Theory based Cooperative Spectrum Sensing in CRNs

Jyotirmay Gupta  
gupta.jtrm@outlook.in

Mekala Manvithasree  
manvithasree@gmail.com

Prakash Chauhan<sup>\*</sup>  
prakashc@tezu.ernet.in

Sanjib K. Deka<sup>†</sup>  
sdeka@tezu.ernet.in

Madhabi Nath  
nath.madhabi42@gmail.com

Nityananda Sarma  
nitya@tezu.ernet.in

Department of Computer Science and Engineering, Tezpur University  
Tezpur, Assam, 784028  
India

## ABSTRACT

Spectrum sensing is a key functionality of Cognitive Radio to detect the availability of Primary User (PU) in a licensed band. However, the local sensing performance of Cognitive Radio users is adversely affected by the issues like multi-path fading, shadowing, and receiver uncertainty problem. These issues can be addressed by Cooperative Spectrum Sensing (CSS). In this work, we have proposed a distributed Cooperative Spectrum Sensing scheme using Coalitional Game theoretic model for Cognitive Radio Networks which contributes to improve the sensing performance in terms of detection probability. The utility function of the game is formulated considering the trade-off between gain and cost during coalition formation. It is also proved that the proposed game model has a non-transferable utility amongst the players and the algorithm terminates in finite time with formation of stable coalitions. A simulation study has been carried out to evaluate the performance of the proposed scheme.

## Keywords

Cognitive Radio (CR), Coalitional Game Theory, Cooperative Spectrum Sensing (CSS)

## 1. INTRODUCTION

The traditional fixed spectrum allocation results in inefficient use of spectrum band [3] leaving unused portions of spectrum called as spectrum holes. To utilize the spectrum holes the cognitive radio (CR) [9]came into existence. CR emerges as a technology allowing accessing the spectrum

opportunistically and dynamically leveraging spectral efficiency. CR enabled nodes are called as secondary users (SU) while licensed user are known as primary users (PU). The SUs and PUs constitute a communication network which is termed as Cognitive Radio Network(CRN). The success of CR depends on how to sense the spectrum efficiently to detect the spectrum holes. Individual sensing performed by SUs suffers due to the issues like multi-path fading, shadowing and receiver uncertainty problem arises in the environment [1]. Cooperative spectrum sensing has emerged as a prominent solution to overcome the issues of individual sensing by exploiting the spatial diversity of SUs. Modeling the framework of CSS gives rise to two questions, first: how to form cooperation among the SUs i.e. what will be the cooperation model, and second: what are the parameters that need to be optimized during cooperation i.e. profit-loss trade-off parameters. The two most popular approaches [1] to model a CSS schemes are: Game theoretic and parallel fusion approach. Game theory has been proven [13] as a suitable mathematical tool for designing a model where dynamic interactions among the players are possible. In this work, we have used Cooperative game theory to model the proposed CSS framework. In [13], authors have proposed a distributed coalitional game theoretic model for CSS in which the utility function is designed by considering detection probability as gain and false alarm probability as the cost during coalition formation. However, the cooperation overhead incurred during coalition formation have not been considered in their work. The merge and split rules has been adopted for coalition which might incur the worst case time complexity [12]. An energy efficient CSS scheme using coalitional game theory has been proposed in [7], which uses a distributed algorithm for coalition formation instead of using existing merges and split rules. The CSS scheme proposed in [4], uses coalitional game theory to improve the sensing accuracy by optimizing cooperation overhead like reporting time and reporting energy. In [2], an efficient CSS scheme is proposed using coalitional game theory by redesigning the cost function in the line of work given in [13] which incorporates parameters like battery power and mobility behavior of SUs. The proposed works in [13, 7, 4, 2] uses a central or head node which is termed as fusion center (FC). The SUs within coalitions report their local sensing decisions to their respective FCs to make a global decision for that coalition.

\*Corresponding author

<sup>†</sup>Corresponding author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

ICDCN '17, January 04-07, 2017, Hyderabad, India

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007759>

But during accumulating the local individual sensing results of the SUs, the reliability of the SUs (i.e. how much accurate the local sensing decision of a SU node with respect to coalitional decision) are not considered into the CSS framework, which may have significant impact on the sensing accuracy. Again considering local sensing results from all the SUs in a coalition might not be helpful during global decision making process since some SUs may have very low sensing accuracy.

To overcome these issues, we have proposed an efficient Cooperative Spectrum Sensing (CSS) scheme using coalitional game theory which considers the reliability of the SUs during coalition formation and global decision making process. We propose a distributed framework for coalition formation which has lower time complexity in comparison to merge and split rule based coalition formation technique. We also proved that the proposed game has non-transferable utility proving alongside that the algorithm terminates in finite time producing stable partition. A simulation study has been carried out to evaluate the performance of proposed CSS scheme.

The rest of the paper is organized as follows: Section 2 describes the system model and assumptions. The game theoretic formulation of the proposed CSS scheme is presented in Section 3. Simulation results and analysis are demonstrated in Section 4 followed by Conclusion and Future Work in Section 5.

## 2. SYSTEM MODEL AND ASSUMPTIONS

An ad-hoc CRN is considered consisting of  $n$  number of SUs and one PU. A slotted time system with a time slot duration  $T$  is assumed in which PU and SUs are synchronized. For simplicity, we consider that  $T$  is divided into two halves: sensing time ( $s$ ) and transmission time ( $t$ ). So,  $T$  can be expressed as  $T = s + t$ . Again, the total energy  $E_i$  spent by a SU  $i$  in slot  $T$  is the sum of sensing energy, transmission energy and idle energy given by  $E_{s,i}$ ,  $E_{t,i}$  and  $E_{id,i}$  respectively.  $E_i$  can be expressed as  $E_i = E_{s,i} + E_{t,i} + E_{id,i}$ . The values of  $E_{s,i}$ ,  $E_{t,i}$  and  $E_{id,i}$  can be computed using the following formulae [8]:

$$E_{s,i} = e_{s,i} F_s b_s \quad (1)$$

$$E_{t,i} = e_{t,i} r_i \quad (2)$$

$$E_{id,i} = e_{id,i} \quad (3)$$

where,  $e_{s,i}$ ,  $e_{t,i}$ ,  $e_{id,i}$ ,  $F_s$ ,  $b_s$  and  $r_i$  represents sensing energy, transmission energy, idle energy, sampling frequency, number of bits per sample and rate of transmission of SU  $i$  respectively. Each SU performs local sensing, which can be done as binary hypothesis testing model as described in [6]:

$$x(t) = \begin{cases} \mathbf{n}(t), & H_0 \\ h(t).s(t) + \mathbf{n}(t), & H_1 \end{cases} \quad (4)$$

where  $H_0$  and  $H_1$  represent the absence and presence of PU in the frequency band.  $x(t)$  is the received signal at SU,  $h(t)$  is the channel gain of the sensing channel,  $\mathbf{n}(t)$  is the zero-mean additive white Gaussian noise (AWGN), and  $s(t)$  is the transmitted PU signal.

For the evaluation of the detection performance, the probability of detection and the probability of false alarm given by  $P_d$  and  $P_f$  respectively can be defined as [5] and expressed

by:

$$P_d = P\{\text{decision} = H_1 | H_1\} = P\{Y > \lambda | H_1\} \quad (5)$$

$$P_f = P\{\text{decision} = H_1 | H_0\} = P\{Y > \lambda | H_0\} \quad (6)$$

where  $Y$  is the decision statistic and  $\lambda$  is the decision threshold.

It is assumed that a reliability level has been assigned to each of the SUs in the network depending on their sensing performances. In this work, a reliability level ranges from 0 to 1 has been assigned to each of the SUs in random manner. It is also considered that all the SUs within a coalition senses same channel at the same time. Each SU performs local sensing using Energy Detection [1] and takes a local decision. The members in each coalition report their local decisions to the respective coalitional heads also known as fusion center (FC). The FCs make global decision and propagate the same to their respective members in the coalition. The individual probability of false alarm and probability of detection given by  $P_{f,i}$  and  $P_{d,i}$  respectively for a SU  $i$  within a coalition  $S$  can be given as [7]:

$$P_{f,i}(S) = Q\left(\frac{\epsilon}{\sigma_n^2} - 1\right)\sqrt{sF_s} \quad (7)$$

$$P_{d,i}(S) = Q\left(\left(\frac{\epsilon}{\sigma_n^2} - \gamma_i - 1\right)\sqrt{\frac{sF_s}{2\gamma_i + 1}}\right) \quad (8)$$

where,  $\epsilon$ ,  $\sigma_n^2$ ,  $\gamma_i$  and  $Q$  represent detection threshold for SUs, noise variance, signal-to-noise ratio and complementary distribution function respectively.

The probability of false alarm and probability of detection for a coalition  $S$  given by:

$$P_{f,S} = 1 - \prod_{j \in S} (1 - P_{f,j}) \quad (9)$$

$$P_{d,S} = 1 - \prod_{j \in S} (1 - P_{d,j}) \quad (10)$$

Let  $P_{H,1}$  is defines as the probability that PU is active, and  $P_{H,0}$  the probability that PU is silent for the time slot in the given channel. Hence,  $P_{H,1} + P_{H,0} = 1$ . Then we can define  $P_{0|0}$  and  $P_{0|1}$  for a coalition  $S$  as:

$$P_{0|0} = P_{H,0}(1 - P_{f,S}) \quad (11)$$

$$P_{0|1} = P_{H,1}(1 - P_{d,S}) \quad (12)$$

where,  $P_{0|0}$  and  $P_{0|1}$  represents probability of PU being absent when PU is actually absent and probability of PU being absent while PU is actually present respectively.

### 2.1 Decision Fusion

The reported decisions to a coalitional head or FC are combined using OR rule for the first level of decision fusion. In the second level, all the FCs communicate with each other to take a global decision. This, again, is done using the OR fusion rule. The decision taken in the second level is the final decision for the whole network and is propagated to all the member nodes of coalitions through their respective FCs.

In the first level a FC, it only considers the decisions of those nodes or FCs whose reliabilities are greater than  $\zeta_h$ ,

where  $\zeta_{th}$  represents the threshold value of the reliabilities. This ensures that during fusion, the decision of those nodes whose reliabilities are at least  $\zeta_{th}$  are considered, these nodes are termed as *strong nodes*, while nodes having  $\zeta < \zeta_{th}$  are considered as *weak nodes*. The aim of this method is to consider only strong nodes during decision fusion activity ensuring the decision taken is reliable and more accurate.

### 3. GAME THEORETIC FORMULATION OF THE PROPOSED DISTRIBUTED CSS

In CSS, SUs form coalitions in order to improve their sensing performance considering their mutual benefit which can be addressed using cooperative game theory. In this work we propose a coalitional game theoretic CSS scheme which uses the following definitions.

**Definition 1:** The set  $S = \{S_1, \dots, S_k\}$  ( where  $k$  is the number of coalitions in the network) is a partition of  $N$ , where  $N$  represents the entire network if  $S_i \cap S_j = \emptyset, \forall (i, j)$   $1 \leq i, j \leq k, i \neq j$  and  $\sum_{i=1}^k (S_i) = N$ .

**Definition 2:** A partition  $S = \{S_1, S_2, \dots\}$  for the  $(N, U)$  coalition formation game is a stable partition if  $S_i$  ( $|S_i| > 0, \forall S_i \in S$ ) does not have any incentive for combining with  $S_j$ , ( $\forall S_j \in S$ ),  $j \neq i, |S_j| < |N|$ .

**Definition 3:**  $S \triangleright W$  implies that aggregate utility of partition  $S$  is greater than aggregate utility of partition  $W$  i.e.,  $\sum_{i=1}^{|S|} U(S_i) > \sum_{i=1}^{|W|} U(W_i)$ .

**Definition 4:** One round of the CSS algorithm is defined as the activity where each fusion center  $fc_i, 1 \leq i \leq k$ , where ( $k = |\hat{FC}|$ ); broadcasts the information sequentially.

#### 3.1 Proposed Game Theory Model

In coalitional game structure, a game  $G$  can be represented as  $(N, U)$ , where  $N$  and  $U$  represent number of players and their utility respectively. Modeling  $U$  for the game  $G$  is discussed in the next section.

##### 3.1.1 The Utility Function

The utility of the game allows the SUs to improve the throughput gain that can be achieved through forming the coalitions with cost incurred in terms of the overhead of energy/time consumption or penalty due to miss detection. We design the utility function based on the ratio of total throughput gain achieved by a coalition to its energy consumption counterpart. Inspired by [7], the utility function  $U(S)$  for a coalition  $S$  can be modeled as:

$$U(S) = \frac{Gain(S) - Cost(S)}{E_{total}(S)} \quad (13)$$

where  $E_{total}(S)$  is the total energy consumed by coalition  $S$  in a time slot  $T$ ,  $Gain(S)$  is the gain achieved by  $S$  in terms of throughput per SU in the coalition  $S$ ,  $Cost(S)$  is the penalty which is related to the  $P_{m,S}$  (miss detection probability) of the coalition  $S$ . The values of  $Gain(S)$ ,  $Cost(S)$  and  $E_{total}(S)$  respectively for a coalition  $S$  are given by:

$$Gain(S) = P_{0|0}(T-s) \sum_{i=1}^{|S|} (\zeta_i r_i) \quad (14)$$

$$Cost(S) = P_{0|1} R_{PU} (T-s) \quad (15)$$

$$E_{total}(S) = E_s | S | s + P_{0|0} E_t (T-s) \\ + P_{1|1} E_i (T-s) + P_{1|0} E_i (T-s) \quad (16)$$

where  $s$ ,  $\zeta_i$ ,  $r_i$  and  $R_{PU}$  represents sensing time, reliability level, data rate of SU  $i$  and data rate of PU respectively. Inspire by [11] and [10], the average throughput  $R_i$  of  $i^{th}$  SU can be reformulated as

$$R_i = \frac{P_{H,0}(1 - P_{f,S})r_i}{P_{m,S} + \beta} \quad (17)$$

where,  $P_{H,0}$  represents the probability of PU being absent. Here  $\beta$  is taken as a very small constant value to compensate the fact that in most of real world scenario, it is impossible to achieve a  $P_{d,S} = 1$ . Hence the value of  $P_{m,S}$  (which is equal to  $(1 - P_{d,S})$ ) can never be 0.

**Property 1:** Proposed coalition formation game has non-transferable utility

**PROOF.** The utility function for the game of the proposed coalition model is given by Equation (13) which is a function of  $Gain(S)$ ,  $Cost(S)$  and  $E_{total}(S)$ . Since the final decision of every SU in a coalition is dictated by the decision made by FC, the probability of false alarm and probability of detection of every SU within a coalition is same as the probability of false alarm and probability of detection of the whole coalition. Therefore the throughput of each SU in a coalition is equal to the throughput of that coalition. As the throughput of coalition  $S$  is not arbitrarily distributed amongst its members, it can be established that the game has non-transferable utility.  $\square$

#### 3.2 The algorithm for FC selection

---

##### Algorithm 1 Algorithm for FC selection

**Input:** Coalitions  $S_i$  and  $S_j$  and their respective FCs

**Result:** New  $fc_{ij}$  if coalition is formed

**Steps:**

1. If  $S_i$  and  $S_j$  combine to form  $S_{ij}, i \neq j$ 
    - (a) Compare  $P_{d,i}$  and  $P_{d,j}$  of  $fc_i$  and  $fc_j$  respectively
    - (b)  $fc_{ij} = fc$ , where  $fc = \max(P_{d,i}, P_{d,j})$
- 

##### 3.2.1 Time complexity of Algorithm 1

The proposed algorithm for Fusion Center selection is a constant time algorithm because we consider every single node as singleton coalition containing itself as a member and acts as FC of its own coalition. We choose a new FC only when two coalitions combine to form a large coalition, which requires only one comparison.

### 3.3 The Distributed CSS Algorithm (DCSS)

Algorithm 2 realizes the proposed coalitional game theoretic CSS model. At the beginning, each of the SUs is considered as singleton coalitions being themselves as FCs of the coalitions.

#### 3.3.1 Time complexity of Algorithm 2

To figure out the time complexity of the algorithm, let there be  $k$  singleton coalitions in  $N$  and  $\forall fc_i \in \hat{FC}$ ,  $fc_i$  is one hop reachable to  $fc_j$ ,  $\forall fc_j \in \hat{FC}$  and  $j \neq i$ . In the first round  $fc_i$ ,  $\forall fc_i \in \hat{FC}$  broadcasts information sequentially. Fusion center  $fc_i$ ,  $\forall fc_i \in \hat{FC}$  compares the combined utility values received from some of the fusion centers in the network. Therefore at the most  $n(n - 1)$  comparisons are done in the first round. In worst case only one new coalition is formed in one round. Let a coalition  $S_i$  combines with  $S_j$  where  $S_i, S_j \in S$  and  $i \neq j$  then  $|S| = n - 1$ . Similarly in the next round at most  $(n - 2)$  comparisons are possible, which implies at most  $(n - 1)(n - 2)$  comparisons. In worst case  $S = \{S_1\}$ ,  $|S_1| = n$  or  $S = \{S_1, S_2\}$ ,  $|S_1| = 1$  and  $|S_2| = n - 1$ . Thereby the time complexity  $T(n)$  is given by :

$$\begin{aligned} T(n) &= n(n - 1) + (n - 1)(n - 2) + \dots + 2 \times 1 \\ &= \frac{(n - 1)(n)(2n - 1)}{6} + \frac{n(n - 1)}{2} \\ &= \frac{n(n - 1)^2}{3} \\ &= O(n^3) \end{aligned}$$

#### 3.3.2 Stability of the DCSS Algorithm

To achieve a stable cooperative decision the proposed CSS model must attain the stability of the game which uses the physical model described in Section 2 for application level requirement. The proposed DCSS algorithm creates stable coalitions and terminates in finite time which in turn leverage the Theorem 1 and 2.

**Theorem 1:** Algorithm 2 terminates in finite amount of time.

**PROOF.** In the worst case a grand coalition is formed, with only one new coalition formation in each round. This requires at most  $n^2$  iterations. Thus this algorithm is assured to terminate in finite time.  $\square$

**Theorem 2 :** Algorithm 2 terminates at a stable partition

**PROOF.** In a particular iteration of a round at most two coalitions can change their utilities. Let  $S_i^{(l)}$  and  $S_j^{(l)}$  be the coalitions which combine at  $l^{th}$  iteration ( $l = 1, 2, \dots$ ) to form a new coalition  $S_{ij}$ . As the combination of  $S_i^{(l)}$  and  $S_j^{(l)}$  to form  $S_{ij}^{(l)}$  occurs if  $U(S_i) + U(S_j) < U(S_{ij})$  and the utilities of all other coalitions remain unchanged in that iteration, the aggregate utility of the coalition in the  $l^{th}$  iteration would either increase or remain unchanged compared to the  $(l - 1)^{th}$  iteration. Thereby every iteration produces sequence of partitions in terms of coalition  $S^{(0)}, S^{(1)}, \dots, S^{(l)}$  such that either  $S^l = S^{(l-1)}$  or  $S^l \triangleright S^{(l-1)}$ . In those cases the Algorithm 2 may lead to two situations:

**Case 1.** A grand coalition is formed. In this state the aggregate utility value is the maximum, as there is no pos-

---

#### Algorithm 2 DCSS Algorithm

---

**Input:**  $P_f$  and  $SNR$  of all SUs and  $\zeta_{th}$

**Result:** CSS decision

**Steps:**

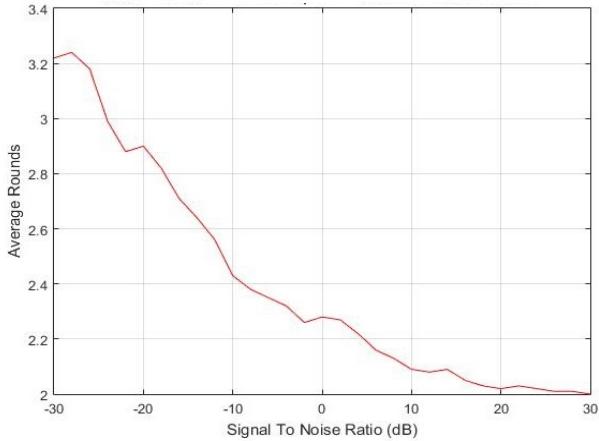
1. Each SU senses the PU channel individually using non co-operative spectrum sensing using energy detection method
  2. Initialisation of coalition formation game
    - (a) Initiate  $r=1, x=0$  go to 2(c)
    - (b) if  $r < |\hat{FC}|$ ,  $r = 1$ , go to 2(c)
    - (c) Fusion center  $fc_i \in \hat{FC}$  broadcasts the information
  3. If fusion center  $fc_j, \forall fc_j \in \hat{FC}, j \neq i$ ,  $fc_j$  is one-hop neighbour of  $fc_i$ 
    - (a)  $fc_j$  calculate  $U(S_{ij})$
    - (b) if  $U(S_{ij}) > U(S_j)$ , send  $U(S_{ij})$  to  $fc_i$
  4.  $fc_i$  receives  $U(S_{ij})$  from  $fc_j, \forall fc_j \in \hat{FC}, i \neq j, fc_j$  is one-hop neighbour of  $fc_i$ ,  $U(S_{ij}) > U(S_j)$   
All the received utility values are stored in a vector  $U^*$ ,  $U^* = [U(S_{im}), U(S_{in}), \dots]$ 
    - (a) Find the maximum of received utility values  $U_{max} = Max(U^*)$
    - (b) If  $U_{max} > U(S_i)$ , form a coalition and send acknowledgement to corresponding fusion center with  $U_{max}$ . Set  $x = 0$  and go to step 5
    - (c) If  $U_{max} < U(S_i)$ , set  $x = x + 1$   
If  $x = n$ , go to step 7 else go to step 6
  5. Update  $\hat{FC}$  by adding  $fc_{ij}$  and removing  $fc_i$  and  $fc_j$  and do necessary updates
  6.  $r = r + 1$  and go to step 2(b)
  7. A stable partition is formed and each SU sends its local sensing decision to its respective fusion center.
  8. Fusion center  $fc_i, \forall fc_i \in \hat{FC}$ , combines the decisions received from SUs of its own coalition having  $\zeta > \zeta_{th}$  and fuses them to make a combined coalitional decision by using OR-rule.
  9. Every fusion center exchanges their coalitional decisions with other fusion centers to make a global decision by applying OR-rule
  10. Every fusion center transmits the final sensing decision to each of the SU within the coalition
- 

sibility to form new coalitions further. Therefore as per Definition 2 it can be stated that  $N$  is stable.

**Case 2.** A partition  $S = \{S_1, S_2, \dots\}$  is formed where  $|S| \neq 1$ . If there are no changes in  $S$  for  $n$  consecutive iterations, i.e., if for a partition  $S^l$ , in any  $k^{th}$  round,  $S^l = S^{(l+1)} = S^{(l+2)} = \dots = S^{(l+(n-1))}$ , it means for each coalition  $S_i$  of the partition  $S$  in  $k^{th}$  round,  $U(S_i) > U(S_j), i \neq j \forall S_j \in S$ . Hence, any non empty coalition  $S_i$  of the partition  $S$  can never increase the aggregate utility of the system. So using Definition 2,  $S = \{S_1, S_2, \dots\}$  will be a stable partition for game.  $\square$

## 4. SIMULATION RESULTS AND ANALYSIS

This section shows the simulation results, which details



**Figure 1:** Average number of rounds to attain stability

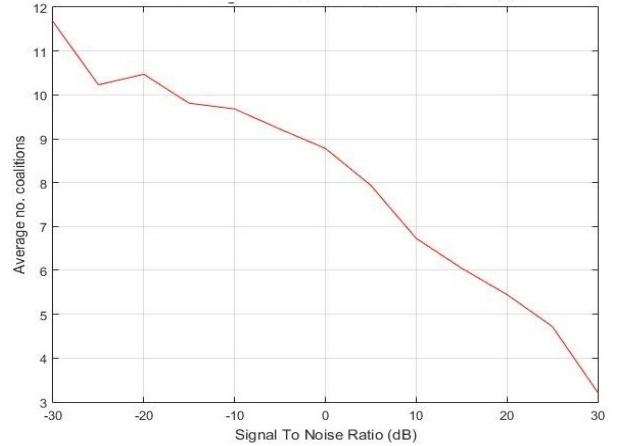
the experiment conducted to show the CRN behavior for the proposed CSS scheme. The results show the variations of average number of coalitions w.r.t. SNR,  $P_d$  w.r.t SNR and  $P_d$  w.r.t.  $P_f$ . The results show how the proposed DCSS model improves sensing performance in comparison to individual spectrum sensing. However the comparison of our proposed technique with existing ones is not feasible due to inconsistency in the method which uses coalitional game theoretic modelling with different criteria such as reliability of the SUs during coalition formation. MATLAB based simulation has been done for performance evaluation. To conduct simulation study for the proposed model, the parameters considered are shown in the Table I.

Parameter	Value
$T$	100 milliseconds
$s$	10 milliseconds
$e_s$	500 nJ per second per bit
$e_{id}$	100nJ per second
Channel bandwidth	1 kHz
$b_s$	16 bits per sample
$P_{H0}$	0.20
$P_{H1}$	0.80
SNR	-50dB to 30db (variable)

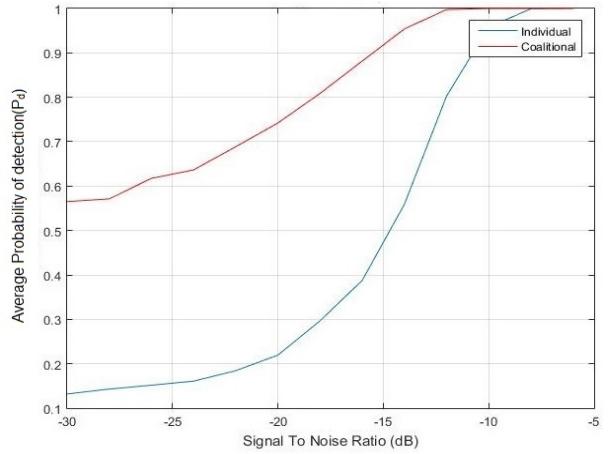
**Table 1:** Parameters used in simulation with their values

Figure 1 represents the average number of rounds to form a stable set of coalitions versus SNR. It shows that The number of rounds to attain stability approaches to 2 for higher values of SNR and become constant thereafter. It can be noted from the graph that at lower values of SNR, it takes more rounds (time) to become stable. This happens because at higher SNR values, SUs have higher values of  $P_d$  and hence are comparatively more stable than they are at low SNR.

Figure 2 gives an idea of how the number of coalitions varies with SNR values. It can be observed that the number of coalitions steadily decreases with an increase in SNR. This



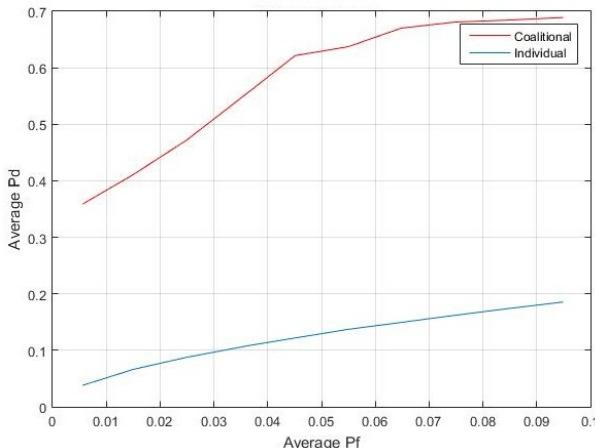
**Figure 2:** Average number of coalitions vs SNR



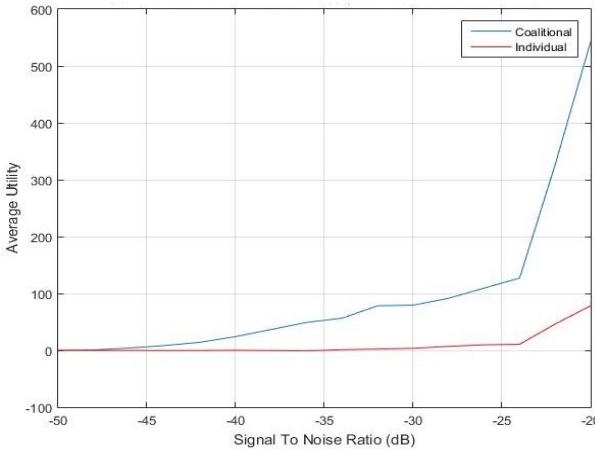
**Figure 3:** Average  $P_d$  vs SNR

trend can be explained by Equation (13). With high SNR values, the SUs have very low  $P_m$ , hence the cost (Equation (15)) decreases impeding the increase in utility each time an SU joins the coalition.

The plots in Figure 3, 4 and 5 show the relation between the parameters like  $P_f$ ,  $P_d$  and utility of each SU with change in SNR. It also shows the advantage of the proposed DCSS model over individual sensing model. From Figure 3 it can be seen that for low values of SNR the gain of each SU, in terms of average  $P_d$  is very high. This happens because at low SNR, the computed  $P_d$  for each SU is very low and they can gain by performing CSS and hence increasing their  $P_d$  values as can be verified from Equation (10). But as the SNR values increase, the gain in terms of probability of detection start decreasing because of the improvement in sensing performance of each SU with better SNR. The maximum gain can be seen in the range of -30dB to -20dB. The Figure 4 demonstrates that the  $P_d$  can be maximised for a given value of  $P_f$ . It can be observed that for a given value of  $P_f$ , the proposed DCSS model can give a higher  $P_d$ . For instance, for a given  $P_f$  value of 0.05, the average individual  $P_d$  can be maximized from 0.137 to 0.638 which



**Figure 4:** Average  $P_d$  vs Average  $P_f$



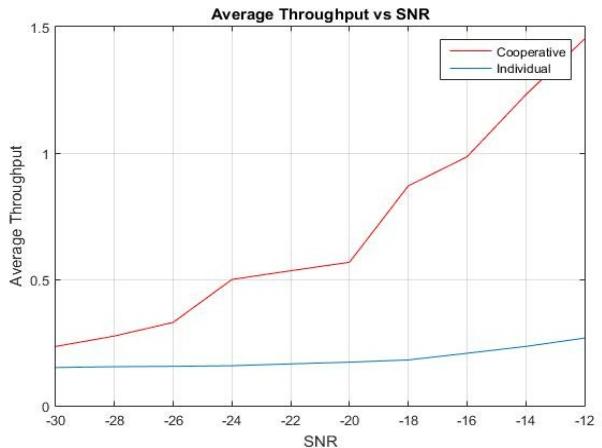
**Figure 5:** Average Utility (individual and cooperative) SU vs SNR

reflects that with an increase in average  $P_f$  both individual and cooperative  $P_d$  get improved.

From the Figure 5 it can been observed that the average utility of a SU has a steady increase from -40dB to -22dB and a steep rise from -23dB to -20dB. It happens because at high SNR, the value of  $P_{d,S}$  increases which in turn improve the utility by reducing cost in terms of penalty from the PU. Figure 6 shows that the average throughput per SU for individual as well as CSS with SNR variations. A Similar explanation can be deduced for the Figure 6 as explained for the Figure 5.

## 5. CONCLUSION AND FUTURE WORKS

In this paper a model for Distributed Cooperative Spectrum Sensing using coalitional game theory is proposed. The proposed model aims at overcoming the drawbacks of individual spectrum sensing. The distributed algorithm incorporates reliability of SUs during coalition formation and decision fusion. The game model creates stable coalitions resulting with non-transferable utility. The simulations based experiment results validate the performance of the proposed



**Figure 6:** Average throughput vs SNR

CSS scheme. Incorporation of scenarios like mobility of SUs and multiple licensed channel availability into the scheme is interesting and is left as a future work.

## 6. ACKNOWLEDGEMENT

The authors would like to thank the Ministry of HRD, Govt. of India for funding as a Centre of Excellence with thrust area in Machine Learning Research and Big Data Analytics for the period of 2014-2019.

## 7. REFERENCES

- [1] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty. Next generation/dynamic spectrum access/cognitive radio wireless networks: a survey. *Computer networks*, 50(13):2127–2159, 2006.
- [2] V. Balaji and C. Hota. Efficient cooperative spectrum sensing in cognitive radio using coalitional game model. In *Contemporary Computing and Informatics (IC3I), 2014 International Conference on*, pages 899–907. IEEE, 2014.
- [3] F. C. Commission. Spectrum policy task force report. *Report ET Docket no. 02-135*, 2002.
- [4] S. K. Deka, P. Chauhan, and N. Sarma. Constraint based cooperative spectrum sensing for cognitive radio network. In *Information Technology (ICIT), 2014 International Conference on*, pages 63–68. IEEE, 2014.
- [5] F. F. Digham, M.-S. Alouini, and M. K. Simon. On the energy detection of unknown signals over fading channels. In *Communications, 2003. ICC’03. IEEE International Conference on*, pages 3575–3579. IEEE, 2003.
- [6] A. Ghasemi and E. S. Sousa. Collaborative spectrum sensing for opportunistic access in fading environments. In *First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005.*, pages 131–136. IEEE, 2005.
- [7] X. Hao, M. H. Cheung, V. W. Wong, and V. C. Leung. A coalition formation game for energy-efficient cooperative spectrum sensing in cognitive radio networks with multiple channels. In *Global*

- Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–6. IEEE, 2011.
- [8] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *System sciences, 2000. Proceedings of the 33rd annual Hawaii international conference on*, pages 10–16. IEEE, 2000.
  - [9] J. M. Iii. An integrated agent architecture for software defined radio. 2000.
  - [10] Z. Khan, J. Lehtomäki, M. Codreanu, M. Latva-aho, and L. DaSilva. Throughput-efficient dynamic coalition formation in distributed cognitive radio networks. *EURASIP Journal on Wireless Communications and Networking*, 2010(1):1, 2010.
  - [11] Y.-C. Liang, Y. Zeng, E. C. Peh, and A. T. Hoang. Sensing-throughput tradeoff for cognitive radio networks. *IEEE transactions on Wireless Communications*, 7(4):1326–1337, 2008.
  - [12] L. Mashayekhy and D. Grosu. A merge-and-split mechanism for dynamic virtual organization formation in grids. *IEEE Transactions on Parallel and Distributed Systems*, 25(3):540–549, 2014.
  - [13] W. Saad, Z. Han, M. Debbah, A. Hjorungnes, and T. Basar. Coalitional games for distributed collaborative spectrum sensing in cognitive radio networks. In *INFOCOM 2009, IEEE*, pages 2114–2122. IEEE, 2009.

# Implementation and Performance Evaluation of Name-based Forwarding Schemes in V-NDN

Divya Saxena

Department of CSE

IIT Roorkee, India

[divya.saxena.2015@ieee.org](mailto:divya.saxena.2015@ieee.org)

Vaskar Raychoudhury

Department of CSE

IIT Roorkee, India

[vaskar@ieee.org](mailto:vaskar@ieee.org)

Christian Becker

Chair for Information Systems II

Universität Mannheim, Germany

[Ls-becker@uni-mannheim.de](mailto:Ls-becker@uni-mannheim.de)

## ABSTRACT

Vehicular networking has recently gained great attention. Although TCP/IP protocol has shown great resilience over the years, still it is quite challenging in highly vehicular environments. Recently, researchers proposed the use of a content-centric approach, Named Data Networking (NDN), for enhancing the efficiency of vehicular ad-hoc networks. NDN uses application generated variable-length, location-independent names to retrieve and/or disseminate the content efficiently. Naming allows NDN applications to access multiple network interfaces at the same time without the need of acquiring IP address repeatedly. In this paper, we implement IP-based data forwarding schemes, such as Epidemic, Spray & Wait, and Adaptive Forwarding using NDN on the sparsely-connected real vehicular testbed. We also evaluate these NDN-based forwarding schemes to study the performance of name based forwarding for retrieving and disseminating data. Our experimental results obtained through real vehicular testbed validate the performance and usability of NDN over VANET.

## Categories and Subject Descriptors

C.2.1 [COMPUTER-COMMUNICATION NETWORKS]: Network Architecture and Design

## Keywords

Named Data Networking, NDN, V-NDN, Interest forwarding, name based forwarding, ICN, CCN.

## 1. INTRODUCTION

In recent years, Vehicular Ad-hoc Networks (VANETs) have emerged as a potential solution for developing a wide range of safety, road traffic, and infotainment applications. Several protocols have been proposed to improve the performance of VANET applications. But still, vehicular communication using TCP/IP is challenging in the highly dynamic environment due to frequently changing topology, short-lived and intermittent connectivity, session management, etc. Moreover, in current Internet, an IP address is required for every networking interface. In other words, as the location changes, new IP address is needed to continue the communication. To handle the afore-mentioned problems, researchers have proposed some opportunistic IP-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*ICDCN '17*, January 04-07, 2017, Hyderabad, India  
 © 2017 ACM. ISBN 978-1-4503-4839-3/17/01 \$15.00  
 DOI: <http://dx.doi.org/10.1145/3007748.3007766>

routing based protocols to utilize the benefits of wireless broadcast nature. But still, IP address assignment and their management were required. Furthermore, Delay Tolerant Networking (DTN) was proposed which allowed late binding of data name to IP address. But still, IP addresses were needed to forward the packets [1].

In order to achieve better performance for the content retrieval and dissemination in a highly dynamic environment, recently, a new networking paradigm, Named Data Networking (NDN) is proposed as a content retrieval and dissemination solution. NDN uses application generated variable-length, location-independent names to retrieve and/or disseminate the content efficiently. Moreover, naming allows NDN applications to access multiple network interfaces at the same time without the need of acquiring IP address repeatedly [1][2]. In NDN, each data packet is self-contained and location independent which allows in-network caching of contents. Further, in-network caching supports consumer mobility. NDN architecture inherently supports the features of DTN and MANET while these are a separate set of solutions in the current Internet. Furthermore, in NDN, devices can communicate if physically they are in the range of each other or along device-to-device paths without having any Internet Service Provider (ISP) [3]. Some researchers have shown that NDN is exponentially growing to provide mobility support in the infrastructure-less networks [4][5][6][7].

Suppose a user  $U$  in vehicle forwards a request ( $R$ ) to retrieve weather-related information over the Wi-Fi network. Vehicles near to  $U$  and along the forwarding path can overhear the corresponding content and opportunistically cache for fulfilling same  $R$ s in future. These vehicles can provide cached information to other vehicles, even without any wireless coverage using Dedicated Short Range Communications (DSRC) [8]. So, NDN can efficiently retrieve and disseminate information in highly mobile environment. Moreover, NDN routers support multi-path forwarding which allows routers to forward  $R$  to multiple interfaces at the same time. There is no specific way to test and evaluate the performance of vehicular data dissemination mechanisms. Most of the existing schemes are either analyzed through simulations or mathematical model to guarantee realistic traffic and driving circumstances. In this paper, we implement IP-based data forwarding schemes such as Epidemic [9], Spray and Wait [10], and Adaptive Forwarding [11] on the sparsely-connected real vehicular testbed using NDN. We also evaluate the performance of these NDN-based forwarding schemes to explore the performance of name based forwarding on data retrieval and dissemination.

The main contributions of this paper are as follows.

- We implement IP-based data forwarding schemes, such as Epidemic, Spray and Wait, and Adaptive Forwarding using

NDN on the sparsely-connected real vehicular testbed.

- We evaluate these NDN-based forwarding schemes to study the performance of name based forwarding on data retrieval and dissemination.
- Our experimental results obtained through real vehicular testbed validate the performance and usability of NDN over VANET.

In the next section, we shall discuss NDN overview and benefits of NDN for VANET.

## 2. BACKGROUND AND MOTIVATION

We first introduce NDN and its applicability to VANET. We also discuss why named data performs well in mobile environment than IP.

### 2.1 NDN Overview

NDN architecture supports pull-based data delivery model in which consumer sends the requests using the content name (CN). In NDN, two packets are used for communication: Interest packet (*Ipkt*) for the request and Data packet (*Dpkt*) for the response. Each *Dpkt* follows the path of its *Ipkt* in reverse. The main aim of NDN is to deliver valid content to requesters without requiring them to know the location of the hosting entity.

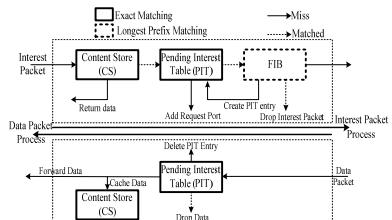


Figure 1. Forwarding process at NDN node [2]

### 2.2 Forwarding in NDN

Each NDN router maintains three main data structures (see Figure 1) to forward *Ipkts*: Content Store (CS), Pending Interest Table (PIT) [12] and Forwarding Information Base (FIB) [13][14]. CS is a temporary cache which stores the content for fulfilling the future requests. PIT maintains the state of each incoming *Ipkts* until they are satisfied. FIB maintains name prefixes and corresponding next-hop(s) to forward *Ipkts* using Longest Prefix Match (LPM).

Whenever an *Ipkt* reaches to a node, its CS is searched and *Dpkt* is forwarded downward if content is available. Otherwise, CN of the *Ipkt* is searched in the PIT and list of incoming interface(s) is updated if an entry for CN already exists in the PIT. If not, *Ipkt* is forwarded upward using FIB and a new entry is created in PIT.

### 2.3 Why NDN is more suitable for VANET?

Current Internet has shown great resilience over the decade, but still, it is suffering from many challenges [15] for developing the VANETs. We discuss the current Internet related shortcomings for VANETs and the solutions supported by the NDN: *First* is, current Internet is based on the end-to-end principle which requires re-establishing TCP connection for resuming the communication. To facilitate communication to multiple network interfaces (Bluetooth, Wi-Fi, etc.) at the same time, current Internet requires different IP addresses for each interface. IP address assignment for each communication disconnection, topology change and network interface makes the Internet complex. NDN handles this issue using the CN for communication which remains consistent across all layers. Naming allows NDN applications to explore multiple network interfaces at the same time and there is no need to establish an

end-to-end connection to the data source for data retrieval. Also, NDN does not maintain sessions. *Second* is, in Internet, whenever host identification fails during the connection establishment or any router at the forwarding path fails, data is lost. NDN supports in-network caching which allows intermediate routers to cache the content to fulfill the future requests for the same content. *Third* is, IP address binds the consumer to the location, e.g., BBC iPlayer service can only be accessed using UK IP address. While NDN uses the CN which is location independent [1].

Instead of these benefits, NDN architecture inherently supports consumer mobility, multi-path forwarding, Interest aggregation, loop detection, congestion control, security, etc.

## 3. RELATED WORK

In recent years, several opportunistic routing schemes [9][10][11] have been proposed for the delay tolerant applications. But, all these schemes suffer from same challenges faced by current Internet as discussed in the sub-section 2.3. To handle aforementioned issues, several researchers [4][5][6][7] introduced NDN for supporting the mobility in the infrastructure-less networks. Moreover, some researchers [16][17] have discussed the feasibility of applying content-centric approach to VANET.

Wang, et al. [5] have proposed in-vehicle architecture to develop the NDN applications for the Vehicle-to-Vehicle (V2V) data dissemination. Each vehicle (V) is embedded with the NDN forwarding engine having the multiple network interfaces. V can send *Ipkts* to the nearby Vs using one hop broadcast. Grassi, et al. [4][6][7] have proposed a vehicular NDN (V-NDN) framework to retrieve data in both infrastructure and infrastructure-less networks. In V-NDN, a vehicle can play the role of consumer, producer, forwarder and data mule (physical carriers of packets). Furthermore, Ming, et al. [18] have performed extensive analysis of handover events in NDN and IP networks w.r.t fast moving trains. The results show that for the first *Ipkt*, NDN performs well as there is no need of any prior registration to new access point.

Several researches have done to explore the forwarding strategies in the wireless ad-hoc networks using NDN. Azgin et al. [19][20] have proposed a framework to evaluate the consumer and producer mobility under the different scenarios. Authors also evaluated the performance of two main widely used forwarding strategies: *flooding* and *smart-flooding*. In flooding, a node broadcasts the content request to all network interfaces while in smart-flooding, request is forwarded to all interfaces when high priority interfaces are not available. Wang et al. [21] have used set of defer timers for minimizing the possibility of collision in the NDN vehicular networks. Grassi, et al. [4] have proposed a greedy forwarding strategy in which consumer request is broadcasted in all directions. Each *Ipkt* appends geolocation of its own vehicle and broadcasts it to all neighbor vehicles. The farthest vehicle among all other vehicles will forward the *Ipkt* further, while rest will do nothing. Amadeo, et al. [22] have proposed that to control flow in the network and to handle the collision, propagation of packets should be limited. If any node N overhears the channel and gets to know that another node has transferred same data multiple times, then N drops that data.

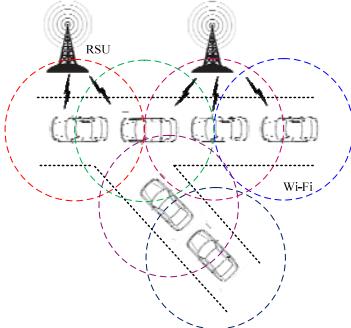
Other studies are also presented for the data collection in NDN VANET, such as naming [23], Pub/Sub integration [24], etc.

## 4. NAME-BASED FORWARDING SCHEMES

Existing VANET information dissemination techniques can be categorized into two models: *push*, and *pull*. The push model is

used to disseminate information proactively while pull model disseminates information on-demand. The push model is used for the safety and emergency applications where immediate response is required while pull model is preferred for delay tolerant applications, such as comfort and entertainment.

NDN architecture is based on the pull model for data dissemination in which consumer explicitly broadcasts the request to get the corresponding content delivered. NDN can also support push model at the network layer in which small data can be included to the *Ipkt*. Generally, pull model has less overhead and low latency in compared to push based model. Pull performs better in terms of efficiency as requests for content are not propagated in parts of the network where no receiver node is located. Pull performs exceptionally well when the numbers of the request receiving nodes are low. In the pull model, consumer interacts less with content producer which makes the consumer more scalable. But, pull based approach is slower in compared to push as a query is broadcasted first and then waits for the reply.



**Figure 2. V2V and V2R communication in VANET**

Figure 2 shows the V2V and Vehicle-to-Roadside (V2R) communication in VANET. It follows the *store-and-forward* standard to retrieve and/or disseminate data from/to network. Each vehicle moves randomly and independently from other vehicles. Therefore, an efficient forwarding scheme is required to forward the *Ipkt* towards the content producer and to fetch corresponding content from multiple carriers of data. We choose Epidemic, Spray and Wait, and Adaptive Forwarding flooding based forwarding schemes to explore the benefits of NDN in the wireless broadcast medium. We implement them on the real vehicular NDN testbed and evaluate the performance. Epidemic, Spray and Wait, and Adaptive Forwarding algorithms are implemented using the NDN and named as *N-Epidemic*, *N-Spray&Wait*, and *N-Adaptive*, respectively.

In *N-Epidemic*, vehicle broadcasts the *Ipkts* to all its neighbors. A neighbor receiving the *Ipkt* acts as a relay node and same process is repeated till *Ipkt* reaches to valid carrier of the data. In VANET, there is no need for *Dpkt* to follow *Ipkt* forwarding path in reverse. *Dpkts* are following the same forwarding scheme followed by *Ipkts*. In *N-Spray&Wait*, a vehicle forwards the *Ipkt* to the first '*N*' vehicles it encounters and then these *N* vehicles carry the *Ipkts* till they get the corresponding content. In *N-Adaptive*, when two vehicles meet, they check the state of the *Ipkt* to be forwarded in their PITs. If a state of *Ipkt* does not exist in another vehicle, then *Ipkt* is forwarded to another vehicle otherwise not forwarded. These algorithms will work in a similar manner for push based approach except the return of the data.

## 5. RESULTS & ANALYSIS

To analyze the performance of name based forwarding in the NDN vehicular network in real time, we have performed a real

testbed analysis on a smart vehicular network. All vehicles in the testbed are sparsely connected and may have intermittent connectivity. In smart NDN vehicular testbed, all vehicles will be embedded with programmed sensors to take necessary decisions.

In this section, we discuss the implementation of the name based forwarding and test it for finding its usability.

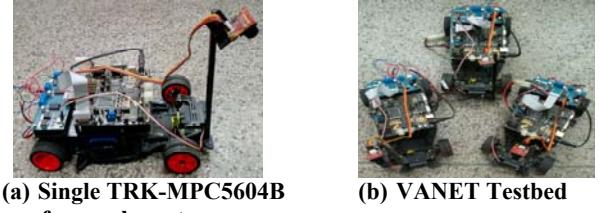
### 5.1 Performance Metrics

Below, we have formally defined the performance metrics to evaluate the name-based forwarding schemes.

- Interest-Data response delay: It is the time taken by an *Ipkt* to reach a node having the requested content and the corresponding *Dpkt* to return requested node.
- Overhead: It is the total number of *Ipkts* forwarded by all nodes of the network for acquiring corresponding content.
- Interest delivery ratio: It is the proportion of the number of *Ipkts* received by the intermediate nodes to total number of *Ipkts* generated by the source and intermediate nodes.
- Network Throughput: It is the rate of successful *Dpkt* delivered per unit time over a communication channel and is represented in *bits per second* (bps).

### 5.2 Experimental Setup

We have developed an NDN vehicular testbed in which smart vehicles move randomly in a specified area and communicate with each other and to the stationary Road Side Units (RSU) (Figure 3). The VANET consists of three TRK-MPC5604B Freescale motor vehicles (car) and four desktop computers (acting as RSUs) inter-communicating through IEEE 802.15.4 protocol.



**(a) Single TRK-MPC5604B free-scale motor car**

**(b) VANET Testbed**

**Figure 3. Prototype Testbed**

The hardware used in the NDN vehicular testbed are MPC5604B-TRK Micro-controller chip, Motor drive board, Servo Motor, Arduino Uno, Wireless Shield, XBee shield, XBee and USB cables. Each motor car and RSU is enabled with the programmed Freeduino board with Arduino wireless SD shield XBee. The motor control board controls the position and movements of vehicle. XBee Module-ZB Series 2, integrated with the Arduino board is used to provide the communication among vehicles. Vehicles exchange messages with other vehicles, when they are in the communication range of each other. For RSUs, we have wired Arduino with a desktop computer. The messages are stored in the 2 GB SD card attached to the Arduino wireless SD shield.

We placed all three cars in such a way that they are not in communication ranges of each other initially. First, we started with only two cars and repeated the procedure for ten times to mitigate the effect of hardware dependency. Then, an average is taken. We increased the number of nodes (vehicles and RSUs alike) one-by-one to analyze the impact of the number of nodes on the network. For each result, same procedure is repeated 10 times and an average is taken. Each vehicle is generating 5 *Ipkts* / sec.

### 5.3 Results

Figure 4 shows the comparison of Interest-Data response delay for three forwarding schemes w.r.to increasing number of vehicles.

The result shows that response delay is decreasing as the numbers of vehicles are increasing in the network because of increasing number of relay nodes. Figure 5 shows the average overhead of forwarding schemes. We can conclude from Figure 4 and 5 that *N-Epidemic* overhead is high in compared to other forwarding schemes while response delay of *N-Epidemic* is low. Interest delivery ratio and average throughput of forwarding schemes are shown in Figure 6 and 7, respectively. We can conclude from the results that the performance of *N-Epidemic* is highest among all schemes and it is performing well in sparsely connected NDN vehicular network w.r.t to delay and throughput while inducing high overhead. In wireless ad hoc network, loss of a packet is quite common. Therefore, to follow the path followed by the *Ipkt* in reverse is not useful. Due to this, it is not necessary to maintain route information in the PIT. In *N-Adaptive*, maintenance of PIT and search for *Ipkt* forwarding degrades the performance in compared to *N-Epidemic*. Moreover, *N-Adaptive* performance is approximately equivalent to *N-Epidemic* while *N-Spray&wait* performance is poor.

## 6. CONCLUSION AND FUTURE WORKS

Recently, researchers proposed the use of content-centric approach, Named Data Networking (NDN), for enhancing the efficiency of vehicular ad-hoc networks. In this paper, we implemented IP-based data forwarding schemes, such as Epidemic, Spray & Wait, and Adaptive Forwarding using NDN on the real vehicular testbed. We evaluated the performance of these NDN based forwarding schemes to examine the performance of name based forwarding on data retrieval and dissemination. Our experimental results obtained through real vehicular testbed validate the performance and usability of NDN over VANET. In future, we will implement more forwarding schemes on the real testbed to a great scale for the different scenarios and applications.

## 7. ACKNOWLEDGEMENT

This work is partially supported by the Alexander von Humboldt Foundation through the post-doctoral research fellow Dr. Vaskar Raychoudhury.

## 8. REFERENCES

- [1] D. Saxena, et al., "Named Data Networking: A Survey," *Elsevier Computer Science Review*, 2016.
- [2] L. Zhang, et al., (2010). Named Data Networking (NDN) Project. [Online]. Available: <http://named-data.net/project/annual-progress-summaries/> [Accessed 10-Sep-2015].
- [3] K. Shilton, et al., A world on NDN: Affordances & implications of the named data networking future Internet architecture, Tech. Rep. NDN-0018 (2014), pp. 1–19.
- [4] G. Grassi, et al., Navigo: Interest forwarding by geolocations in vehicular named data networking, 2015.
- [5] L. Wang, et al., Data naming in vehicle-to-vehicle communications, *INFOCOM WKSHPS*, 2012, pp. 328–333.
- [6] G. Grassi, et al., Vehicular inter-networking via named data *ACM SIGMOBILE Mob. Computer Comm.*, 17 (3) (2013), pp. 23–24.
- [7] G. Grassi, et al., VANET via Named Data Networking, *INFOCOMM Workshop*, 2014, pp. 410–415.
- [8] J.B. Kenney DSRC stds. in U.S. Proc., 99 (7) (2011), pp. 1162–82
- [9] Vahdat A, Becker D. Epidemic routing for partially connected ad hoc networks. Tech. Rep. CS-200006. 2000
- [10] Spyropoulos T, et al., Spray and wait: an efficient routing scheme for intermittently connected mobile networks. *SIGCOMM workshop on delay-tolerant networking*; 2005. p. 252–9.
- [11] Hou F, Shen X. An adaptive forwarding scheme for message delivery over delay tolerant networks. *GLOBECOM 2009*.
- [12] D. Saxena and V. Raychoudhury, "Radient: Scalable, Memory Efficient Name Lookup Algorithm for Named Data Networking", *JNCA*, Volume 63, Jan., 2016.
- [13] D. Saxena and V. Raychoudhury, "N-FIB: Scalable, Memory Efficient Name-based forwarding", *JNCA*, September, 2016.
- [14] D. Saxena, et al., "Reliable Memory Efficient Name Forwarding in Named Data Networking", *EUC*, 2016, Paris, France.
- [15] G. Tyson, et al., A Mauthe A survey of mobility in information-centric networks, *ACM Commun.*, 56 (12) (2013), pp. 90–98
- [16] G. Arnould, et al., A self-organizing content centric network model for hybrid vehicular ad-hoc networks, *DIVANet*, 2011, pp. 15–22.
- [17] P. Talebifard, et al., A content centric approach to dissemination of information in vehicular networks, in Proc. of the DIVANet, 2012, pp. 17–24.
- [18] Z. Ming, et al., Efficient handover in railway networking via named data, *Int. J. Mach. Learn. Cybern.* (2014), pp. 1–7
- [19] A. Azgin, R. Ravindran, G. Wang, A scalable mobility-centric architecture for named data networking, 2014.
- [20] A. Azgin, et al., Mobility study for named data networking in wireless access networks, *ICC*, 2014, pp. 3252–3257.
- [21] L. Wang, et al., Rapid traffic information dissemination using named data, *MobiHoc workshops*, 2012, pp. 7–12.
- [22] M. Amadeo, et al., Enhancing content-centric networking for vehicular environments, *Comp. Netw.*, 2013, pp. 3222–34
- [23] J. Wang, et al., DMND: Collecting data from mobiles using named data, *VNC*, 2010, pp. 49–56.
- [24] W. Drira, et al., A Pub/Sub extension to NDN for efficient data collection and dissemination in V2X networks, *WoWMoM*, 2014, pp. 1–7.

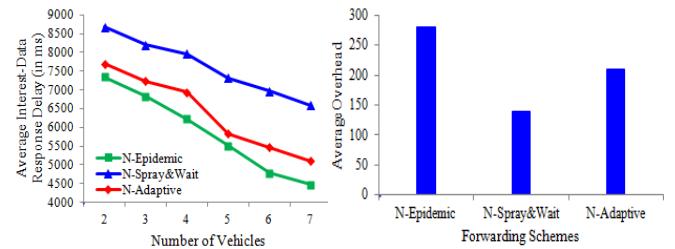


Figure 4. Average response delay between *Ipkt* and *Dpkt*

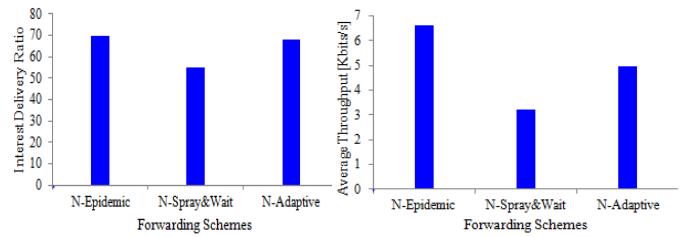


Figure 5. Average overhead of different forwarding schemes

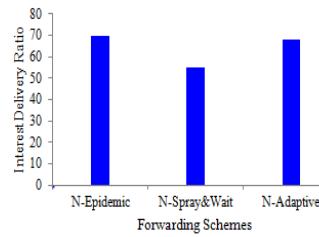


Figure 6. Interest delivery ratio

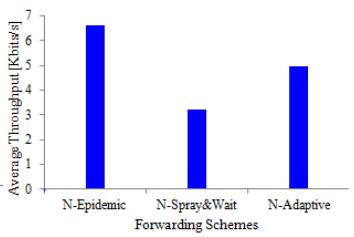


Figure 7. Throughput

# A Planning Based Approach for Context Aware Services Composition in Pervasive Systems

[Short Paper]

**Sujata Swain**  
 Department of Computer Science and  
 Engineering  
 Indian Institute of Technology Roorkee  
 Uttarakhand, India-247667  
 sujataswain019@gmail.com

**Rajdeep Niyogi**  
 Department of Computer Science and  
 Engineering  
 Indian Institute of Technology Roorkee  
 Uttarakhand, India-247667  
 rajdpfec@iitr.ac.in

## ABSTRACT

Due to the proliferate use of mobile devices, the Internet is now easily accessible and such ubiquitousness helps users by offering services anytime, anywhere. Typically, a complex user request is satisfied by composing several services. In pervasive systems the set of services may change as the context changes. In this paper, we model context-aware services composition problem as a planning problem. Some salient aspects of our approach are: (i) a user gives request in her native language, which is then compiled to a planning domain description language (PDDL) and (ii) the environment is monitored and a plan is generated according to the situation. We have implemented a context-aware education system to illustrate our approach.

## CCS Concepts

- Human-centered computing → Ubiquitous computing;
- Computing methodologies → Natural language processing; Planning for deterministic actions;

## Keywords

Automated planning; Context-aware services composition

## 1. INTRODUCTION

Recent technological advances in handheld devices, sensors, and communication networks have transformed the vision of pervasive systems (made in the early 1990s by Mark Weiser [10]) into a reality today. Pervasive systems are embedded with a context-aware component that automatically senses the environment and adapts itself to the changes in the environment without user intervention.

Context-aware services composition is the problem of finding a composed service for a given request when a context changes. Context refers to some information about a user

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

*ICDCN '17, January 04-07, 2017, Hyderabad, India*

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007764>

and/or the environment. Consider a situation where a user, at some point of time, may be working in a laptop in her office and at another point of time she would be using a smart phone while in a car. Here context is a location (office or car) and when the context changes the devices associated with the context change which in turn may lead to changes in services.

In this paper we consider an education scenario where different types of study package (full course, crash course, and summary course) pertaining to a course is provided depending on the location of the student (which may be home, car, or classroom) and the number of days remaining for the examination. We model this scenario as a planning problem [8]. A student may give the request in her native language which is then compiled to a planning domain description language (PDDL). We wish to obtain plans that satisfy the goal for different contexts.

The rest of this paper is organized as follows: related work is discussed in Section 2. The proposed approach is given in Section 3. The context-aware education system is given in Section 4. We conclude the paper in Section 5.

## 2. RELATED WORK

A classification framework of context-aware system is suggested in [1]. The framework consists of the following five layers: concept and research layer, network layer, middleware layer, application layer, and user infrastructure layer. In context-aware setting, a request may not be satisfied due to some services that are not available. In [2], an unavailable service is replaced by functionally equivalent services. A context-aware plan architecture and a hybrid approach to build a plan corresponding to a context-aware service composition is proposed in [3], based on global planning and local optimization solution. An OWL-S service composition planner, called OWLS-Xplan is suggested in [4] which converts OWL-S services to an equivalent PDDL description. It invokes the Xplan planner to generate a plan satisfying a given goal. A context framework that facilitates the development and deployment of context-aware adaptable Web services is given in [5]. A framework for automated composition of Advanced Telecom Services for environmental early warnings is proposed in [6]. However, monitoring the environment is not discussed in [6].

### 3. PROPOSED APPROACH

#### 3.1 Architecture

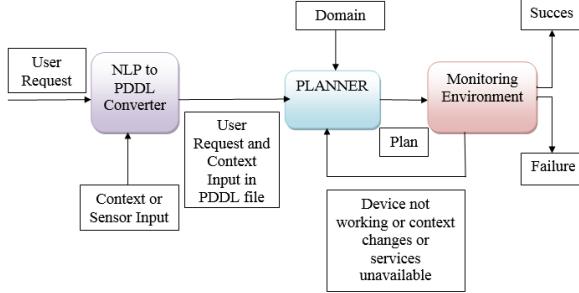


Figure 1: Architecture of System

The architecture of proposed system is shown in Figure 1. It has three important modules: (a) NLP to PDDL, planner, and environment monitoring.

##### NLP to PDDL Module

The module receives a request from a user in her native language which is then compiled to an equivalent PDDL description. It has four sub-tasks: (a) natural language processing (NLP), (b) context information processing, (c) device ranking, (d) domain selection.

NLP [7] take as input sentences and produces as output structured representations that captures the meaning of these sentences. These involves, part-of-speech tagging, chunking, named entity recognition and semantic role labeling.

Context information processing: Context information is collected through sensors. Context includes location of user, devices available to the user, network available to the user and user's activity.

Device ranking: The devices are ranked according to screen size, memory, and network bandwidth.

Domain selection: Domain repository contains the PDDL files of all domains. The goal determines a domain PDDL file according to matching function.

##### Planner Module

A planning problem is a tuple  $\langle P, O, I, G \rangle$  where,  $P$  is a finite set of propositions,  $O$  is a finite set of actions,  $I \subseteq P$  is the initial state and  $G \subseteq P$  is a goal state.

Planner module takes two inputs: problem PDDL file and domain PDDL file. Domain file contains a set of actions which are defined by preconditions and effects. An action is executed only when the precondition is satisfied and it produces effects. The planner module generates a plan (composed service) which contains a set of actions to be executed.

##### Monitoring Environment Module

This module monitors the environment. It detects the following: change in location, device is not working, unavailability of a service.

#### 3.2 Algorithm

The algorithm given below aims to satisfy the goal  $G$  anywhere and anytime, even if a context changes, that is for all possible contexts  $c$  in a given domain, find a plan  $\pi$  such that  $\pi \models_c G$ . This is shown in Figure 2.

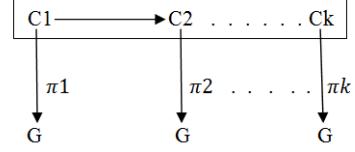


Figure 2: Goal Maintenance With Context Change

It takes as input an initial location  $l$  of the user and the request  $R$  in the native language of the user. From the request, verbs and nouns are obtained using the POSTagger function [9]. Using the verbs and nouns, we find the goal  $G$ . In the next step, domain description is obtained from the domain repository using goal  $G$ . Initial state  $I$  is obtained from the given location  $l$  and request  $R$ . A planning problem is generated as a tuple  $\langle P, O, I, G \rangle$ . The planner module takes as input a planning problem and generates a *plan* (composed service) using a classical planner (Black-box). The monitoring environment function will capture changes in the environment for a fixed time period. Depending upon the changes in the environment, the algorithm tries to find the corresponding plan. Failure is returned when no plan is found. This may be due to the unavailability of services and/or devices.

---

#### Algorithm 1

```

1: Input: Request in native language  $R$ , Initial location  $l$ 
2: Output: plan or failure
3: find verb and noun from  $R$  using POSTagger function.
4: find goal  $G$  using verb and noun.
5: select Domain description  $(P, O)$  using goal  $G$  from domain repository.
6: rank the devices available to the user.
7: generate initial state  $I$  from  $l$  and  $R$ .
8: plan := find_plan( $P, O, I, G$ )
9: if plan = nil then
10:   return failure;
11: monitorEnvironment();
12: if location changes from  $l$  to  $l'$  then
13:   generate  $I'$  from  $l'$  and  $R$ ;  $I := I'$ 
14: if device  $d$  is not working then
15:   replace  $d$  by the next rank device  $d'$ ;
16:   generate  $I'$  from  $d'$ ;  $I := I'$ 
17: if service  $s$  is unavailable then
18:   replace  $s$  by  $s'$ 
19: goto Line 8.
  
```

---

### 4. CONTEXT-AWARE EDUCATION SYSTEM

Our proposed Context-Aware Education System (CA-ES) is an application that helps learning process of a student and saves time and effort for notes preparation. We assume an education system in which a student enrolls for a course, attends the lectures, and prepares notes with the help of lectures and books. Thereafter she appears in the examination. CA-ES provides the study material to the student based on the context. Here, two contexts are considered: student's location and number of days remaining for the examination.

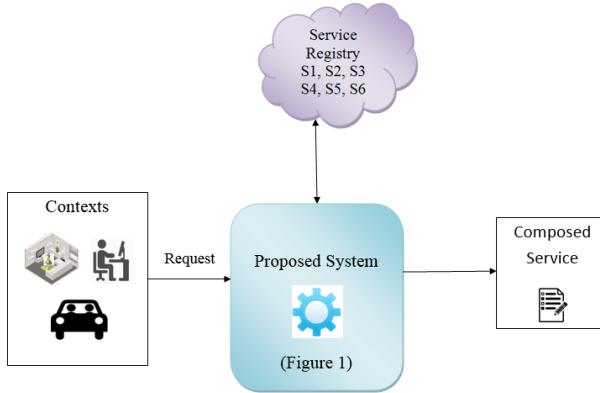
The CA-ES delivers lectures based on the number of days remaining for the examination and the best device available at the student's location. The structure of this system is shown in Figure 3.

A student gives the date of the first examination and the CA-ES determines the number of days remaining based on the current date. Let  $X$  be the number of days remaining before the first examination. Let course duration be 5 months (150 days approximately).

The CA-ES provides a course material  $\alpha$  where

$$\alpha = \begin{cases} Full\_course\_material; & if \quad 30 < X \leq 150 \\ Crash\_course\_material; & if \quad 2 < X \leq 30 \\ Summary\_course\_material; & if \quad 0 < X \leq 2 \end{cases}$$

CA-ES senses the available device(s) at the student's location to deliver the study material. The device selection is based on the device ranking. CA-ES ranks the devices based on screen size, network bandwidth, and other information of the devices. According to the rank and availability status (may be busy for other work) of the device, CA-ES provides services to the best device.



**Figure 3: CA-ES System**

For example, let  $S_1, S_2, S_3, S_4, S_5$ , and  $S_6$  be the services for providing course material, where:

$S_1$  or  $S_2$ : provide full course material

$S_3$  or  $S_4$ : provide crash course material

$S_5$  or  $S_6$ : provide summary course material

Let the devices available at the student location (home) be SmartTV, Desktop, Laptop, Tablet, E-reader, Smartphone. The ranking of the devices will be: i. SmartTV, ii. Laptop, iii. Desktop, iv. Tablet, v. Smartphone, vi. E-reader.

## 4.1 Implementation details

We have implemented the context-aware services composition problem as a planning problem. The experiments were run on the Intel core i5 with 2.53 GHz machine with 4 GB of RAM. We have selected the PDDL files which run on a classical planner e.g., Blackbox. We have implemented the CA-ES system as described above. The details are given below.

Let the request send by a student be “Want to study Geography”.

### NLP to PDDL Module:

This module receives the request from the student. It translates the request from native language into English. However, in this case, the request is in English. So, there is no need of translation. Otherwise, it invokes a language translation web service to translate the native language to English. By using NLP techniques [9], this module gets the selective information from request, i.e., main verb and noun.

In this request, main verb is *study* and noun is *Geography*. It generates a goal state from the above information i.e.,

(:goal (study Geography))

It selects the domain from the domain repository by matching the goal state to the predicates (facts) that are available in the domain PDDL file. The predicates are given below:

```

(:predicates (user ?u)(location ?x)
(device ?y)(at ?u ?x)(in ?y ?x)(ON ?y)
(subject ?sub)(study ?sub)(crashcourse ?sub)
(videocrashcourse ?sub)
(Tv ?y)(Laptop ?y)(Radio ?y) (Ereader ?y)
(Video ?y) (Audio ?y) (Text ?y) )
  
```

It receives the context information of the student through GPS (global positioning system) and records the student's current location and current date. It calculates the number of days remaining for the examination and finds which type of service is required. Let the number of days remaining be 20. Then it requires *crash course* material of *Geography*. Let current location be *home* and devices available to the student are *SmartTV*, *Laptop*, and *Mobile phone*. Let the devices be ranked according to the screen size. The ranking of the devices will be: i. SmartTV, ii. Laptop, iii. Smartphone.

It generates the initial state from the context information processing and device ranking. Smart TV is selected for device as it has the highest ranking.

```

(:init (user sujata) (Tv tv)
(location Home)(at sujata Home) (in tv Home)
(subject Geography)(crashcourse Geography)))
  
```

### Planner module:

It obtains the planning problem from NLP to PDDL module and generates a plan. The actions are:

```

SwitchOnTv (Sujata Home tv tv)
videoCrashCoursesubject(Geography tv tv)
Provide-CrashCourse (Geography)
  
```

*SwitchOnTv* is an action whose precondition is TV should be present at the student's location and effect is to switch on the TV.

*videoCrashCoursesubject* is an action whose effect is Videocrashcourses.

*Provide-CrashCourse* is an action whose precondition is calling of videocrash service and it's effect is study Geography which is the goal predicate.

The actions (in PDDL) required for satisfying the above request are given below.

```
(:action SwitchOnTv
:parameters (?u - user ?curpos - location
?y - device ?z - Tv)
:precondition (and (user ?u) (location ?curpos)
(at ?u ?curpos)(device ?y) (Tv ?y)
(in ?y ?curpos))
:effect ((ON ?y)))

(:action videoCrashCoursesubjectservice
:parameters (?sub - subject ?y - device
?y - Video )
:precondition (and (subject ?sub)
(crashcourse ?sub)(device ?y)
(Video ?y ))
:effect (videocrashcourse ?sub ))

(:action Provide-CrashCourse
:parameters ( ?sub - subject )
:precondition (and (subject ?sub)
(videocrashcourse ?sub ))
:effect (study ?sub ) )
```

#### **Monitoring Environment Module:**

It monitors the environment by sensing the changes in the context and then it generates a new plan corresponding to the new context.

##### *Case-1: Location changes*

Suppose the location of the student changes from home to car. CAES application detects the change of location and obtains the new initial state corresponding to the new context.

Let the devices available at the car be Radio, Smart phone, E-reader. According to the current location, it gives ranks to available devices and let Radio be selected. Now the new initial state is:

```
(:init (user sujata) (Radio radio)
(location Car)(at sujata Car) (in radio Car)
(subject Geography)(crashcourse Geography)))
```

The new plan consists of the following actions.

```
SwitchOnRadio (Sujata Home radio)
AudioCrashCoursesubject (Geography radio)
Provide-CrashCourse (Geography)
```

##### *Case-2: Unavailability of some services*

Suppose that service S3 for providing the crash course material be unavailable. The planner module select a functional equivalent service (in this case S4) and generates a new plan.

##### *Case-3: Some of the devices are not working*

Suppose that the SmartTv is not working. The CAES selects the next rank device (i.e. Laptop) and obtains the new initial state and generates a new plan.

The initial state and the actions of the plan are given below.

```
(:init (user sujata) (Laptop Laptop)
(location Home)(at sujata Home) (in Laptop Home)
(subject Geography)(crashcourse Geography)))
```

```
SwitchOnLaptop (Sujata Home laptop)
VideoCrashCoursesubject (Geography laptop)
Provide-CrashCours (Geography)
```

## 5. CONCLUSION

We modeled a context-aware services composition problem as a planning problem where a user gives request in her native language, which is then compiled to a planning domain description language (PDDL) and the environment is monitored and a plan is generated according to the new context. We developed an algorithm to generate a plan (composed service) corresponding to a given context. An important feature of the algorithm is that it monitors the environment by sensing changes in the context. We developed a context-aware education system that integrates the proposed algorithm. As part of our ongoing/future work, we would like to come up with an implementation that provides an interface for sensing the real-world context changes.

## 6. REFERENCES

- [1] J. Y. Hong, E. Suh, and S. J. Kim. "Context-aware systems: A literature review and classification," *Expert Systems with Applications*, vol. 36, no. 4, pp: 8509-8522, 2009.
- [2] S. Swain and R. Niyogi. "An Ontology Based Approach for Satisfying User Requests in Context Aware Settings," In 30th IEEE International Conference on Advanced Information Networking and Applications (AINA), Crans-Montana, pp. 1130-1137, 2016.
- [3] L. Qiu, Z. Shi, and F. Lin. "Context Optimization of AI planning for Services Composition," In IEEE International Conference on e-Business Engineering (ICEBE'06), Shanghai, pp. 610-617, 2006.
- [4] M. Klusch, A. Gerber, and M. Schmidt. "Semantic web service composition planning with owl-s-xplan," In Proceedings of the AAAI Fall Symposium on Agents and the Semantic Web, pp. 55-62, 2005.
- [5] M. Keidl, and A. Kemper. "Towards context-aware adaptable web services," In Proceedings of the 13th International World Wide Web conference, pp. 55-65, ACM, 2004.
- [6] A. Ordóñez, V. Alcazar, J. C. Corrales, and P. Falcarin. "Automated context aware composition of Advanced Telecom Services for environmental early warnings," *Expert Systems with Applications*, vol. 41, no. 13, pp: 5907-5916, 2014.
- [7] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, pp. 2493-2537, 2011.
- [8] M. Ghallab, D. Nau, and P. Traverso. "Automated planning: theory & practice," Elsevier, 2004.
- [9] Part-Of-Speech Tagger (POS Tagger) : <http://nlp.stanford.edu/software/tagger.shtml>
- [10] M. Weiser, "The Computer for the 21st Century," *Scientific American*, September, 1991.

# Exploring the Impact of Connectivity on Dissemination of Post Disaster Situational Data over DTN

Suman Bhattacharjee  
IIT, Kolkata  
Kolkata, INDIA  
[suman.bhattacharjee@heritageit.edu](mailto:suman.bhattacharjee@heritageit.edu)

Siuli Roy  
IIT, Kolkata  
Kolkata, INDIA  
[siuli.roy@heritageit.edu](mailto:siuli.roy@heritageit.edu)

Sukumar Ghosh  
The University of Iowa  
Iowa City, USA  
[sukumar.ghosh@uiowa.edu](mailto:sukumar.ghosh@uiowa.edu)

Sipra DasBit  
IEST, Shibpur  
Kolkata, INDIA  
[sdasbit@yahoo.co.in](mailto:sdasbit@yahoo.co.in)

## ABSTRACT

Opportunistic or Delay-tolerant networks (DTNs) may be used as a viable option for exchanging situational information during post-disaster communication in the absence of existing communication infrastructure. As opposed to traditional communication networks, situational information dissemination in DTNs depends on the degree of intermittent connectivity among the mobile nodes. Higher degree of connectivity implies smaller delay to disseminate situational information which results in better knowledge sharing. The degree of intermittent connectivity is influenced by mobility pattern of the nodes which is heterogeneous in nature. In this work, we attempt to formulate an empirical relationship between the degree of intermittent connectivity and extent of situational knowledge dissemination over DTNs in order to quantify the impact of the node mobility on the inter-shelter knowledge transfer. In the absence of prior knowledge about the mobility patterns, we consider only the broad features of the impact. We introduce two metrics, *Total Encounters* and *Knowledge Sharing Ratio*. *Total Encounters* depends on node density, average movement speed of the nodes and elapsed time after initial node deployment. It serves as an indicator for estimating the degree of intermittent connectivity. The extent of situational information dissemination over an entire network is measured in terms of *Knowledge Sharing Ratio*. The relationship between *Total Encounters* and *Knowledge Sharing Ratio* is validated using the real mobility traces of volunteers captured from a field trial carried out in a disaster affected area. This analysis also helps us to formulate a suitable strategy for deployment of volunteer nodes in a post-disaster scenario.

## CCS Concepts

- Networks → Network performance evaluation → Network performance modeling

## Keywords

Post-Disaster Communication; Delay Tolerant Networks; Node Density; Knowledge Sharing; Node Deployment;

## 1. INTRODUCTION

A well-managed disaster recovery operation requires effective sharing of situational information among relief and rescue

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [Permissions@acm.org](mailto:Permissions@acm.org)

ICDCN '17, January 04-07, 2017, Hyderabad, India  
© 2017 ACM. ISBN 978-1-4503-4839-3/17/01 \$15.00  
DOI: <http://dx.doi.org/10.1145/3007748.3007767>

workers. Such information can help the relief and rescue agencies to appropriately coordinate, manage and channelize their resources [1-2]. Since large parts of the communication infrastructure may get completely incapacitated as a consequence of the disaster, it becomes difficult to collect situational data from the different remote and inaccessible parts and plan the relief and rescue operation accordingly.

A viable option for exchanging situational awareness information is to use Delay-tolerant networks (DTNs), where battery-powered wireless devices (e.g. smart phones, PDAs, laptops, etc.) with multiple communication interfaces (Wi-Fi and Bluetooth), configured in ad-hoc mode, are used to exchange situational information [3].

The situational information is time varying in nature. As a result, the relevance of any situational information is valid only for a certain period of time after which that information becomes stale. Owing to the intermittent and short lived contact duration, timely dissemination of situational information is a challenge in DTNs. Unlike traditional infrastructure-based network, in DTNs such dissemination is purely dependent on the degree of intermittent connectivity among the mobile nodes. The degree of intermittent connectivity is governed by mobility pattern of the nodes. In a post-disaster scenario, mobility patterns of the nodes are heterogeneous and event driven in nature and therefore, hard to model analytically [4]. So, it is very important to quantify the impact of the node mobility on the inter-shelter knowledge transfer. As it is difficult to get prior knowledge about the mobility patterns of nodes, we identify a metric (*Total Encounters*) to measure the degree of intermittent connectivity based on the broad features of the impact of mobility. An increase in the *Total Encounters* among nodes increases the probability of delivery of situational knowledge to far away shelters. Hence, better knowledge sharing is achieved. The extent of situational information dissemination over an entire network is measured in terms of *Knowledge Sharing Ratio*. *Knowledge Sharing Ratio* is influenced by *Total Encounters* that is governed by several factors like node density over an area; average movement speed of the nodes and total time elapsed since node deployment.

Substantial amount of research has been conducted to measure the impact of connectivity on data dissemination in DTNs. However, to the best of our knowledge a suitable strategy for deploying DTN nodes in a post-disaster scenario for effective data dissemination is nonexistent.

Whitbeck et al. [5] modeled the connectivity pattern in the intermittently connected dynamic network as an edge-Markovian dynamic graph and proposed a closed-form expression that links delivery ratio to common intermittently connected mobile network (ICMN) parameters like message size, maximum tolerated delay, and link lifetime. Qiu et al. [6] modeled the

message propagation process based on edge-Markovian dynamic graphs and provided a detailed expression of average information dissemination delay based on message size, users' selfishness and the number of involved subscribers.

Although [5-6] modeled the connectivity pattern of nodes as edge-Markovian dynamic graphs, Pebreyre et al. [7] revealed that under real mobility traces the model contradicted its underlying hypothesis.

These limitations have motivated us to explore the characteristics of opportunistic network connectivity and subsequently establish an empirical relationship between the degree of intermittent connectivity and extent of knowledge dissemination. This relationship helps the disaster management authorities to plan volunteer deployment in disaster affected areas.

The rest of the paper is organized as follows. Section 2 describes the system model. The analysis of Knowledge Sharing Ratio is described in Section 3. Section 4 demonstrates the empirical modeling of node deployment plan. Finally, conclusions are drawn in Section 5.

## 2. SYSTEM MODEL

We consider Delay Tolerant Networks (DTNs) of mobile nodes, where intermittent connectivity occurs due to node mobility. The node mobility is responsible for data dissemination among the nodes. In this work, we assume each shelter generates and shares (broadcasts) situational information periodically to all the other shelters. Figure 1 represents a framework for post-disaster situational knowledge dissemination in a scenario consisting of four shelters. Each shelter is assumed to have a set of tokens that need to be shared with the other shelters via the mobile nodes. The total number of tokens exchanged depends on the degree of intermittent connectivity among all the nodes in the network. In this work the degree of intermittent connectivity is represented in terms of *Total Encounters* ( $E$ ). The mobile nodes move around in an arbitrary pattern with a certain speed where the patterns and the speeds depend on the disaster scenario. After a certain time, a fraction of tokens generated from one shelter reaches other shelters. The fraction of tokens that reach the other shelters from a given shelter determines how much knowledge has been shared from that node.

We measure the amount of dissemination of situational information over the entire network in terms of *Knowledge Sharing Ratio* ( $k$ ). It is defined as the ratio of the average number of situational messages received by all the shelters present in the network to the total number of situational messages generated at different shelters in the network over a period of time. Although Delivery ratio is a well-known metric to measure the amount of data dissemination in DTNs, we have not used it in our work as Delivery ratio measures the amount of data dissemination in unicast mode of communication only. The situational information is disseminated in broadcast mode. Therefore, we introduce the metric *Knowledge Sharing Ratio* in order to measure the amount of dissemination of situational information.

$$k = \frac{\text{Average number of situational messages received}}{\text{Total number of situational messages generated}}$$

*Knowledge Sharing Ratio* is influenced by the overall degree of intermittent connectivity. In our work, the degree of intermittent connectivity in a DTN is measured in terms of *Total Encounters* among all the nodes in the network.

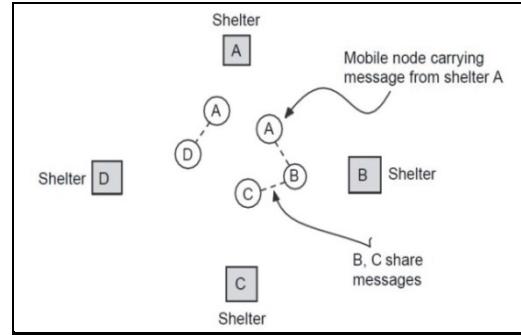


Figure 1. Framework for post-disaster situational knowledge dissemination

The *Total Encounters* ( $E$ ) can be defined as

$$E = \sum_{t=1}^n c_t \quad (1)$$

where  $c_t$  = Number of contacts with duration  $t$ ,  $t$  = duration of each contacts,  $n$  = number of all possible discrete contact durations

## 3. ANALYSIS OF KNOWLEDGE SHARING RATIO

This section presents an analysis of *Knowledge Sharing Ratio* with an objective to find out the parameter(s) which have a significant influence on it. The extent of knowledge dissemination can be predicted by monitoring these parameter values.

As mentioned in Section 1, when the degree of connectivity increases, the extent of data dissemination also increases. Thus, an increase in *Total Encounters* among the nodes increases the probability of delivery of situational knowledge to far-away shelters. As a result, overall *Knowledge Sharing Ratio* ( $k$ ) increases. Therefore, from the above discussion we can write

$$k = f_1(E) \quad (2)$$

*Total Encounters* is defined by Equation (1) in Section 2 in terms of the number of contacts ( $c_t$ ). Moreover,  $c_t$  is dependent on the mobility characteristic of the nodes which is movement speed and movement direction. Post-disaster mobility is purely event driven; therefore, it is practically impossible to predict the movement direction of nodes. So in this work, we assume that  $c_t$  depends on the average movement speed of the nodes denoted by  $s$ . Apart from average movement speed ( $s$ ),  $c_t$  is also dependent on node density ( $n$ ) and elapsed time ( $T$ ). If node density  $n$  in a region is increased by keeping  $T$  and  $s$  as constants,  $c_t$  increases. Similarly,  $c_t$  increases with  $T$  by keeping  $n$  and  $s$  constants. Hence, we can represent  $c_t$  as a function of  $n$ ,  $T$ , and  $s$  and accordingly  $E$  can also be represented as,

$$E = f_2(n, T, s) \quad (3)$$

Substituting  $E$  in relationship (2) we can write

$$k = f_3(n, T, s) \quad (4)$$

The derived relationship in Equation (4) gives us a guideline about node deployment.

## 4. MODELING THE DEPLOYMENT PLAN

In this section, we formulate a mathematical hypothesis between *Total Encounters* and *Knowledge Sharing Ratio*. As mentioned in Section 3, *Total Encounters* is influenced by three parameters ( $n$ ,  $T$  and  $s$ ). So, hypotheses between *Total Encounters* and those three parameters are also formulated. In order to formulate the hypotheses, we observe the dataset captured by simulating a snapshot of the recent earthquake that occurred in Nepal with a magnitude of 7.8 on 25<sup>th</sup> April 2015. The simulation environment is configured using the information received from several sources.

The hypotheses are validated through dataset obtained from the real mobility traces of volunteers captured from a field trial carried out at a super cyclone (Aila) affected area in Namkhana (West Bengal, India) on 27<sup>th</sup> and 28<sup>th</sup> February 2016.

### 4.1 Simulation Setup

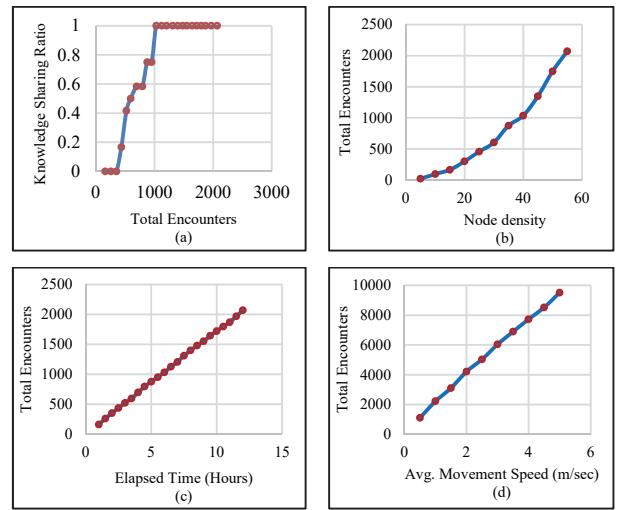
In this work, we perform two different sets of simulation for hypotheses formulation and validation respectively. We run the simulation by varying simulation time and node density. We assume that each shelter broadcasts situational messages to other shelters at a regular interval, say twice a day and that information is disseminated throughout the entire region under simulation. Disaster relief and rescue operations are generally carried out for 12 hours in a day during 6 a.m. to 6 p.m. So, twice a day actually represents one message in every 6 hours.

Initially simulation environment is configured based on a snapshot of the recent earthquake that occurred in Nepal. We create our simulation set-up in ONE simulator [8], version 1.5.1-RC2, based on information provided by the unofficial crisis map regarding post-disaster rescue and relief operation carried out in Nepal, at the Kathmandu area. We consider post-disaster mobility model (PDM) [9], for our simulation.

The second set of simulation environment is configured based on the captured mobility traces of volunteers from a field trial carried out at a super cyclone (Aila) affected area (Namkhana, West Bengal, India). We conducted a mock field trial at Chandanpuri village of Namkhana Block, West Bengal, India where four shelters were set up within an area of 3.5 square Km centered around the Chandanpuri Ramkrishna Ashram. Thirty seven volunteers, equipped with mobile handheld devices, participated in the field trial. We captured the real mobility traces of twenty nine volunteers. The mobility traces of eight volunteers could not be captured due to limited network coverage. Therefore, we use four shelters along with the real traces of twenty nine volunteers in ONE simulator, in order to validate our hypotheses.

### 4.2 Hypotheses formulation

We analyze the simulation results obtained from Nepal earthquake. Figure 2(a) indicates the influence of *Total Encounters* on *Knowledge Sharing Ratio*. During the initial period of simulation, nodes move around in a random pattern with a certain speed. The encounters among the nodes during this period replicate the situational information from different shelters to other neighboring nodes. Consequently, the *Knowledge Sharing Ratio* is not enhanced, as situational information does not reach the intended destinations. In the next phase, nodes accumulating situational information gradually reach the shelters, which enhance the overall *Knowledge Sharing Ratio*. From this point onwards, the graph in Figure 2(a) takes off and finally attains a steady state when all the situational information is delivered to their intended destinations. Hence, the characteristic graph of *Knowledge Sharing Ratio* with respect to the *Total Encounters*



**Figure 2. Interrelationship among (a) Total Encounters vs. Knowledge Sharing Ratio (b) Node Density vs. Total Encounters (c) Elapsed Time vs. Total Encounters and (d) Average movement speed vs. Total Encounters in Nepal Disaster**

represents the characteristics of a logistic curve (“S” shape). The characteristic graph in Figure 2(a) exhibits “S” shape, as well. So

our hypothesis for this scenario is  $k = \frac{L}{1+e^{-\lambda(E-M)}}$  where,  $k = \text{Knowledge Sharing Ratio}$ ,  $E = \text{Total Encounters}$ ,  $L$  = the curve’s maximum value. In this case  $L$  would be 1 as maximum value of  $k$  is 1,  $\lambda$  = the slope of the curve and  $M$  = value of  $E$  at the curve’s midpoint. So the relationship can be rewritten as

$$k = \frac{1}{1+e^{-\lambda(E-M)}} \quad (5)$$

Figure 2(b) represents the impact of growing node density on *Total Encounters* during the Nepal disaster. From the figure, it is clear that *Total Encounters* increases exponentially with increasing node density. So we formulate hypothesis

$$E = a_1 n^{b_1} \quad (6)$$

where,  $n$  = node density,  $a_1$  and  $b_1$  are constants. Similarly, Figure 2(c) represents the influence of elapsed time on the *Total Encounters*. Our hypothesis for this scenario is

$$E = a_2 + b_2 T \quad (7)$$

where,  $T$  = elapsed time,  $a_2$  and  $b_2$  are constants.

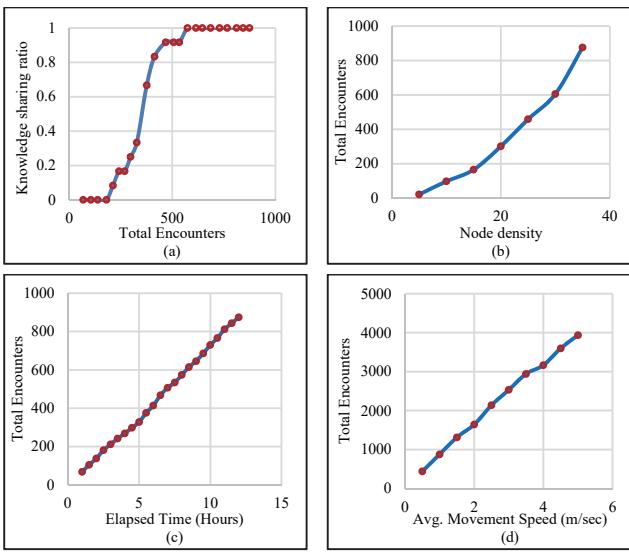
Figure 2(d) represents the influence of average movement speed on the *Total Encounters*. So, our hypothesis for this scenario is

$$E = a_3 + b_3 s \quad (8)$$

where,  $s$  = average movement speed of nodes,  $a_3$  and  $b_3$  are constants.

### 4.3 Hypotheses validation and parameter estimation

In this section, we validate our proposed hypotheses on the dataset collected by simulating the real traces of volunteers captured from



**Figure 3. Interrelationship among (a) Total Encounters vs. Knowledge Sharing Ratio (b) Node Density vs. Total Encounters (c) Elapsed Time vs. Total Encounters and (d) Average movement speed vs. Total Encounters in Namkhana Field trial**

the field trial carried out at Namkhana. Figure 3(a) indicates the influence of *Total Encounters* on *Knowledge Sharing Ratio* and it represents similar growth characteristics like Figure 2(a). Therefore, our hypothesis in Equation (5) is validated. Fitting an appropriate curve on the dataset obtained from Figure 3(a), we get

$$k = \frac{1}{1 + e^{-0.02(E-348)}} \quad (9)$$

The root mean square error (RMSE) and percentage error (PE) is calculated based on the data set obtained from Figure 3(a) to find out the accuracy of the approximated value of  $k$ . We obtain the RMSE = 0.0286 and PE = 6.05% for  $k$ .

Figure 3(b) represents the effect of node density on the *Total Encounters* in the field trial. The visual pattern exhibits identical growth characteristics like Figure 2(b). Thus, the hypothesis mentioned in Equation (6) is validated. Similarly, the patterns represented in Figure 3(c) and Figure 3(d) validates our hypothesis in Equation (7) and Equation (8) respectively. Now, by applying the Least Square method on the data set obtained from Figure 3(b), Figure 3(c) and Figure 3(d) we estimate the values of  $(a_1, b_1)$ ,  $(a_2, b_2)$  and  $(a_3, b_3)$  respectively. We observe that the estimated errors are within the acceptable limit (3.4% to 5.7%). Combining Equations (6), (7) and (8) we get

$$E = ua_1n^{b_1} (a_3 + b_3s)(a_2 + b_2T) \quad (10)$$

where,  $u$  is a constant. We compute the values of  $u$  based on dataset obtained from the field trial at Namkhana and get the range of  $u$  as  $2.226 \times 10^{-6} \leq u \leq 2.338 \times 10^{-6}$ .

We consider the maximum value of  $u$  for computation. Now, by substituting the values of  $E$  and  $u$  in Equation (9) we get,

$$k = \frac{1}{1 + e^{-0.02\{2.338 \times 10^{-6} \times a_1n^{b_1}(a_3 + b_3s)(a_2 + b_2T) - 348\}}} \quad (11)$$

Equation (11) represents a relationship between  $k$ ,  $n$ ,  $T$  and  $s$ .

## 5. CONCLUSION

In this work, we propose a relationship between the degree of intermittent connectivity and extent of situational knowledge dissemination over DTNs. This relationship enables the disaster management authorities to deploy the adequate number of volunteers (DTN nodes) in disaster affected areas in order to accumulate a comprehensive overview of the situation within an acceptable delay. We validate the relationship by dataset obtained from real mobility traces of volunteers captured from a field trial carried out at Namkhana Block, West Bengal, India. The results demonstrate the accuracy of our model. The variation in the perception gap of the individual volunteer associated with each shelter may be studied as a part our future work.

## 6. ACKNOWLEDGMENTS

The work is undertaken as part of Information Technology Research Academy (ITRA), Media Lab Asia project entitled “Post-Disaster Situation Analysis and Resource Management Using Delay-Tolerant Peer-to-Peer Wireless Networks”.

## 7. REFERENCES

- [1] Schulz, A. and Paulheim, H. 2013. Mashups for the Emergency Management Domain. *Semantic Mashups: Intelligent Reuse of Web Resources*; Springer, Berlin, 237-260.
- [2] Endsley, M. R. 1988. Design and evaluation for situation awareness enhancement. In *Proceedings of the Human Factors Society*, vol. 32, 97-101.
- [3] Fall, K., Iannaccone, G., Kannan, J., Silveira, F. and Taft, N. 2010. A Disruption-Tolerant Architecture for Secure and Efficient Disaster Response Communications. In *Proceedings of the ISCRAM*.
- [4] Picu A. and Spyropoulos, T. 2015. DTN-meteo: forecasting the performance of DTN protocols under heterogeneous mobility. *IEEE/ACM Transactions on Networking*, vol. 23, no. 2, 587-602.
- [5] Whitbeck, J. and Conan, V. 2011. Performance of opportunistic epidemic routing on edge-Markovian dynamic graphs. *IEEE Trans. Comm.*, vol. 59, no. 5, 1259-1263.
- [6] Qiu, L., Hui, P., Jin, D., Su, L. and Zeng, L. 2012. Edge-Markovian Dynamic Graph Based Performance Evaluation for Delay Tolerant Networks. In *Proceedings of the IEEE WCNC*, 2129-2133.
- [7] De Pebeyre, A.F., Tarissan, F. and Sopena, J. 2013. On the relevance of the edge-Markovian evolving graph model for real mobile networks. In *Proceedings of the Wireless Days*, 1-6.
- [8] Keranen, A., Ott, J. and Karkkainen, T. 2009. The ONE simulator for DTN protocol evaluation. In *Proceedings of the SIMUTOOLS*. DOI=<http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5674>
- [9] Uddin, M. Y. S., Nicol, D. M, Abdelzaher, T.F. and Kravets, R. H. 2009. A post-disaster mobility model for delay tolerant networking. In *Proceedings of the Winter Simulation Conference*, 2785-2796.

# Disaster-aware WDM network design for data centres

Saja Al Mamoori  
 School of Computer Science  
 University of Windsor  
 Windsor, Ontario N9B 3P4,  
 Canada  
 almamo@uwindsor.ca

Arunita Jaekel  
 School of Computer Science  
 University of Windsor  
 Windsor, Ontario N9B 3P4,  
 Canada  
 arunita@uwindsor.ca

Subir Bandyopadhyay  
 School of Computer Science  
 University of Windsor  
 Windsor, Ontario N9B 3P4,  
 Canada  
 subir@uwindsor.ca

## ABSTRACT

Natural disasters and deliberate attacks may cause large scale damage to communication networks. Such disasters affect specific geographic areas so that, when a disaster occurs, a set of geographically close network components (e.g., nodes and links) may fail simultaneously. It is crucial to develop a robust communication scheme to handle requests for communication, that includes provisions to handle disasters. In this paper we have proposed an optimal approach to the problem of developing robust schemes for communication to handle requests for communication in data center (DC) networks. We have formulated our problem as an Integer Linear Program (ILP) which minimizes the network resources needed to develop such robust schemes. We have studied our approach using simulations, varying parameters, such as the number of DCs, the number of disasters, and the optical reach.

## 1. INTRODUCTION

Schemes to handle faults in optical networks have received a lot of attention. Most schemes consider single link failures only - the most common type of component failure in wide-area backbone networks. *Path Protection* and *link protection* are popular schemes to handle faults, due to its guaranteed fault recovery and fast recovery time. Two approaches for path protection has been studied widely - *Dedicated Path Protection* (DPP) and *Shared Path Protection* (SPP) [1–3]. In dynamic lightpath allocation using path protection, when processing a request for communication from  $S$  to  $D$  is received, the objective is to search for resources to set up two lightpaths - a primary lightpath and a backup lightpath. The routes used by the primary and the backup lightpaths are mutually fiber-disjoint. If the system is able to handle the request for communication (i. e., the search is successful), the primary lightpath is set up and used unless there is an edge failure in the route used by the primary lightpath. Depending on the type of path protection (e.g., 1+1 dedicated path protection, 1:1 dedicated path protection or

shared path protection), the backup lightpath is set up beforehand or is set up if needed. When a failure occurs that affects the path used by the primary lightpath, the backup lightpath is set up, if needed, and data communication resumes, using the backup lightpath. The single link failure model was extended to the concept of *shared risk link group* (SRLG). SRLG was originally proposed to deal with conduit cuts, but it can be extended to include general risks.

A *data center* (DC) can be visualized as a server used for storage, computing resources, as well as distribution of large amounts of data. A network of a number of data centers is often called a *cloud* that provides services to customers [4,5]. Typically the contents and/or services of data centers are replicated over multiple data centers, so the user requests can be served by any DC that hosts the required content. This approach also allows the design of robust systems. Large-scale disasters, natural (e.g., the 2012 Hurricane Sandy, the 2011 Japan Tsunami) or malicious attacks, may affect both the primary and backup resources and/or cause failures at data center nodes. Resilient network design to handle such disasters is now a popular research area [4,6–8]. Most of the work consider the problem of *static lightpath allocation*.

In response to a request for communication, a lightpath (henceforth called the *primary lightpath*) will be set up to handle the communication when the network is fault-free. At the same time, provisions will be made to handle disasters that disrupt the primary lightpath. Lightpaths that will be set up when a disaster affects the primary lightpath will henceforth be called *backup lightpaths*.

The rest of the paper is organized as follows. In Section II we have presented our problem definition and have proposed an ILP to solve it. In Section III we have discussed the simulation results obtained using the proposed approach. We have given our concluding remarks in Section IV.

## 2. AN OPTIMAL ALGORITHM TO SOLVE THE DISASTER-AWARE DYNAMIC RWA

### 2.1 Problem statement

The problem is to handle, if possible, a request for transmitting some file  $f_i$  to a node  $t$  requesting this file. If we are successful in handling this request, we should be able to obtain details about the lightpath (henceforth called the *primary lightpath*) which will be used to handle the fault-free situation and, for each disaster  $d \in \mathcal{D}$ , details about the lightpath (henceforth called the *backup lightpath* to handle disaster  $d$ ) to be used that avoids any edge affected by dis-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

*ICDCN '17, January 04-07, 2017, Hyderabad, India*

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3007765>

aster  $d$ . When processing a request for transmitting file  $f_i$  to a node  $t$ , it is convenient to visualize a new *virtual node*  $s$  and some new *virtual edges* from  $s$  as follows. For each data centre  $\mathcal{S}_i^j$ ,  $1 \leq j \leq m$  containing a copy of the requested file  $f_i$ , we visualize a single virtual edge from virtual node  $s$  to data centre  $\mathcal{S}_i^j$ .

Our objective is to minimize the cost of the resources needed to handle the new request for communication. The resources used to handle the request will be measured by the total number of channels, used to handle this request, which were not used by any of the existing communication.

## 2.2 Notation used in the ILP

We will use the following constants and variables:

**c1, c2:** constants whose values may be adjusted to suit design requirements.

**N:** the set of nodes.

**E:** the set of directed edges of the network. If  $i$  and  $j$  are nodes of the fiber network (including the data centers), edge  $(i, j) \in E$  represents a fiber from node  $i$  to node  $j$ . Set  $E$  also includes the virtual edges from the virtual node  $s$  to each data centre data centre  $\mathcal{S}_i^j$  that contains a copy of file  $f_i$ .

**$d_{max}$ :** the optical reach.

**$\ell_{ij}$ :** the length of the fiber  $(i, j) \in E$ .

**D:** the predefined set of dominant disasters.

**K:** set of channels per fiber.

**$\mathcal{E}^d$ :** the set of edges disrupted due to disaster  $d \in \mathcal{D}$ .

**$c_{ij}^k$ :** a constant for all channel  $k \in K$  and edge  $(i, j) \in E$  where  $c_{ij}^k = 1$  if if channel  $k$  is used on edge  $(i, j)$  either for a primary lightpath or for a backup lightpath to handle some disaster; 0 otherwise.

**$p_{ij}^{kd}$ :** a constant where  $p_{ij}^{kd} = 1$  if if channel  $k$  on edge  $(i, j)$  cannot be used to handle disaster  $d$ ; 0 otherwise.

**$e_{ij}^d$ :** a constant where  $e_{ij}^d = 1$  if edge  $(i, j) \in \mathcal{E}^d$ , 0 otherwise.

**$x_{ij}$ :** a binary variable where  $x_{ij} = 1$  if edge  $(i, j)$  is used by the primary lightpath; 0 otherwise.

**$y_{ij}^d$ :** a binary variable where  $y_{ij}^d = 1$  if edge  $(i, j) \in E$  is used by the backup lightpath to handle disaster  $d$ ; 0 otherwise.

**$z^d$ :** a binary variable for all disaster  $d \in \mathcal{D}$  where  $z^d = 1$  if if the primary lightpath uses any edge disrupted by disaster  $d$ ; 0 otherwise.

**$\mu^k$ :** a binary variable where  $\mu^k = 1$  if channel  $k$  is used by the primary lightpath; 0 otherwise.

**$\nu^{kd}$ :** a binary variable where  $\nu^{kd} = 1$  if channel  $k$  is used by the backup path to handle disaster  $d$ ; 0 otherwise.

**$r_{ij}^k$ :** a bounded variable, where  $r_{ij}^k = 1$  if channel  $k$  on edge  $(i, j)$  is not used in any existing communication, and is used by the new communication; 0 otherwise.

**$q_{ij}^k$ :** a bounded variable where  $q_{ij}^k = 1$  if the primary lightpath uses channel  $k$  on edge  $(i, j)$ , 0 otherwise.

**$w_{ij}^{kd}$ :** a bounded variable where  $w_{ij}^{kd} = 1$  if edge  $(i, j)$  and channel  $k$  are used by the backup path to handle disaster  $d$ , 0 otherwise.

## 2.3 Integer Linear Program Formulation

**Objective:** Minimize

$$c1 \cdot \sum_{k \in K} \sum_{j:(i,j) \in E} r_{ij}^k + c2 \cdot \sum_d z^d \quad (1)$$

**Subject to:**

$$\sum_{j:(i,j) \in E} x_{ij} - \sum_{j:(j,i) \in E} x_{ji} = \begin{cases} 1 & \text{if } i = s, \\ -1 & \text{if } i = t, \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

$$\sum_{j:(i,j) \in E - \mathcal{E}^d} y_{ij}^d - \sum_{j:(j,i) \in E - \mathcal{E}^d} y_{ji}^d = \begin{cases} z^d & \text{if } i = s, \\ -z^d & \text{if } i = t, \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

$$\sum_{j:(i,j) \in E} \ell_{ij} \cdot x_{ij} \leq d_{max} \quad (4)$$

$$\sum_{j:(i,j) \in E} \ell_{ij} \cdot y_{ij}^d \leq d_{max} \quad \forall d \in \mathcal{D}. \quad (5)$$

$$c_{ij}^k \cdot x_{ij} + \mu^k \leq 1 \quad \forall (i, j) \in E, k \in K \quad (6)$$

$$\sum_k \mu^k = 1 \quad (7)$$

$$p_{ij}^{kd} \cdot y_{ij}^d + \nu^{kd} \leq 1 \quad \forall (i, j) \in E, k \in K, d \in \mathcal{D} \quad (8)$$

$$\sum_k \nu^{k,d} = z^d \quad \forall d \in \mathcal{D} \quad (9)$$

$$z^d \geq e_{ij}^d \cdot x_{ij} \quad \forall (i, j) \in \mathcal{E}^d, d \in \mathcal{D} \quad (10)$$

$$z^d \leq \sum_{i,j:e_{ij}^d=1} x_{ij} \quad \forall d \in \mathcal{D} \quad (11)$$

$$q_{ij}^k \leq x_{ij} \quad \forall k \in K, (i, j) \in E \quad (12)$$

$$q_{ij}^k \leq \mu^k \quad \forall k \in K, (i, j) \in E \quad (13)$$

$$q_{ij}^k \geq x_{ij} + \mu^k - 1 \quad \forall k \in K, (i, j) \in E \quad (14)$$

$$w_{ij}^{kd} \leq y_{ij}^d \quad \forall k \in K, (i, j) \in E, d \in \mathcal{D} \quad (15)$$

$$w_{ij}^{kd} \leq \nu^{kd} \quad \forall k \in K, (i, j) \in E, d \in \mathcal{D} \quad (16)$$

$$w_{ij}^{kd} \geq y_{ij}^d + \nu^{kd} - 1 \quad \forall k \in K, (i, j) \in E, d \in \mathcal{D} \quad (17)$$

$$r_{ij}^k \geq q_{ij}^k \quad \forall k \in K, (i, j) \in E \quad (18)$$

$$r_{ij}^k \geq w_{ij}^{kd} \cdot (1 - c_{ij}^k) \quad \forall k \in K, (i, j) \in E, d \in \mathcal{D} \quad (19)$$

$$r_{ij}^k \leq q_{ij}^k + \sum_{d \in \mathcal{D}} w_{ij}^{kd} \cdot (1 - c_{ij}^k) \quad \forall k \in K, (i, j) \in E \quad (20)$$

$$r_{ij}^k \leq 1 \quad \forall k \in K, (i, j) \in E \quad (21)$$

Constraint (2), (3) enforces, for all node  $i \in N$  and all disaster  $d \in \mathcal{D}$ , flow conservation on the path used by the primary lightpath and, for each disaster  $d \in \mathcal{D}$  that disrupts the primary lightpath, the path used by the (backup lightpath) that avoids  $d$ . Constraint (4) ((5)) Ensure that the length of the route used by the primary lightpath and, for each disaster  $d \in \mathcal{D}$ , the (backup lightpath) that avoids  $d$ , is less than the optical reach  $d_{max}$ . Constraints (6) and (7) ensure that exactly one channel  $k$  is used for the primary lightpath. For each disaster  $d \in \mathcal{D}$ , if disaster  $d$  involves an edge used by the primary lightpath, constraints (8) and

(9) ensure that exactly one channel  $k$  is used by the backup lightpath that avoids  $d$ . If the primary lightpath uses (does not use) any edge affected by disaster  $d$ , constraints (10) and (11) will set  $z^d$  to 1 (0). If the primary lightpath uses (does not use) channel  $k$  on edge  $(i, j)$ , constraints (12)-(14) set  $q_{ij}^k$  to 1 (0). If the backup lightpath to handle disaster  $d$  uses (does not use) channel  $k$  on edge  $(i, j)$ , constraints (15)-(17) set  $w_{ij}^{kd}$  to 1 (0). Finally, constraints (18)-(21) set  $r_{ij}^k$  to 1 if either  $q_{ij}^k = 1$ , or  $w_{ij}^{kd} = 1$  for any disaster  $d$  and channel  $k$  is currently unused by all existing communication (i.e.,  $c_{ij}^k = 0$ ), Otherwise constraints (18)-(21) set  $r_{ij}^k$  to 0.

### 3. EXPERIMENTAL RESULTS

In this section, we present our simulation results for evaluating the performance of the proposed strategy under different disaster scenarios. We have considered two well-known physical topologies - the 14-node NSFNET and the 20-node ARPANET networks. We assume that copies of the relevant content (i.e. files) have been distributed at selected data centres, based on some pre-defined replication strategy. Each result reported is obtained by taking the average over 5 simulation runs. The experiments were carried out on a Intel Core i7-3537U CPU 2.50 GHz processor using IBM ILOG CPLEX version 12.6.2 [9].

The simulations were carried out in 2 phases. During Phase I, we investigate how the resource utilization varies, when establishing a set of connection requests, under different disaster scenarios. During Phase II, we assume the network has a set of ongoing connections already established. We randomly generate new connection requests and report the blocking probabilities for different conditions. In the following subsections, we discuss our results for each phase.

Figs. 1 and 2 shows the resource usage for 20 connection requests, over the 14-node and 20-node network topologies, respectively. We assumed there are 8 available channels per fiber (i.e.  $|K| = 8$ ) and a replication strategy with 2 (3) data centers at nodes 1 and 13 (2, 5 and 9). The number of disasters varied from 0 (fault-free case) to 14 (there may potentially be a failure at any node in the network). We see that the number of wavelength-links needed to accommodate the connections, increases monotonically, with the number of disasters and roughly doubles as we go from the fault free case to 14 disasters. Also, it is clear that fewer resources are needed, when we increase the number of data centers. This is expected because, with more data centers available, it is more likely that the file can be found closer to the requesting node. On an average, in most cases 30% - 35% more resources are used, for the same number of disasters, if only two data centers are available, compared to a situation where three data centers were available.

Fig. 3 shows how the resource usage varies with the optical reach. At first the results seem a bit counter-intuitive, since they show that a more restrictive (lower) value of optical reach actually improves resource utilization, i.e. less resources are required for lower values of optical reach. This is because, for lower optical reach values, more copies of the files are required, in order to meet the requirement that at least one copy is reachable from all other nodes. Since more copies of a file are available, distributed over different data center locations, it is more likely that the requesting node is closer to the location where a particular file is stored. This results in a shorter path lengths and lower resource usage (in terms of wavelength links). However, if other types

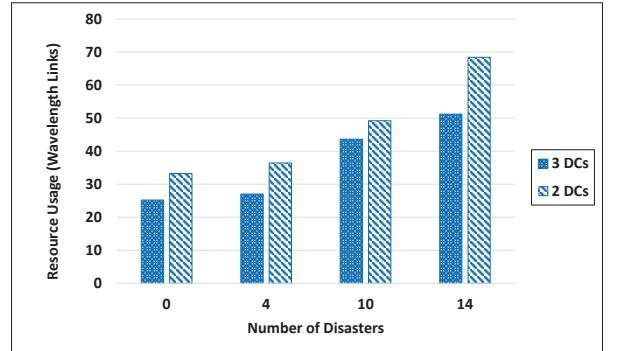


Figure 1: Comparison of resource utilization versus number of disasters for a 14-node network.

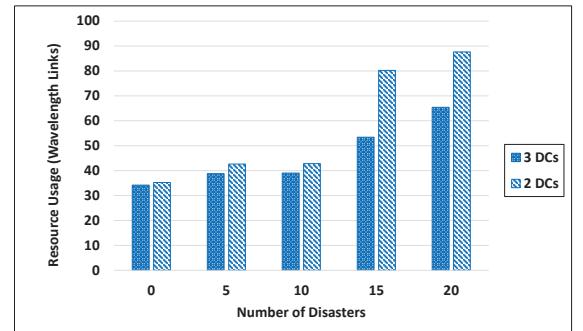


Figure 2: Comparison of resource utilization versus number of disasters for a 20-node network.

of resources, such as number and capacity of data centers, are taken into consideration then lower optical reach values would require more resources.

Table 1 shows a summary of the results of our simulations. As expected, with more traffic on the network (i.e. higher number of initial connections established during Phase I), the blocking probability increases consistently. However, it is interesting to note there was no significant difference in blocking probabilities(BP) for Category I and Category II disasters. This indicates that it is possible to handle additional disasters with relatively little increase in resource usage. On the other hand, the number and location of the data centers seemed to be a more important factor affecting the blocking probability, particularly for the 14-node topology. The table also shows the average resource utilization (RU) and the average ILP solution times for processing a single request for communication. We computed the averages over all the different test cases we studied for that network. As in Phase I, the resource utilization (measured by the number of wavelength links) decreases slightly with the number of data centers.

### 4. CONCLUSIONS

In this study we have proposed a provisioning scheme to handle requests for communication that takes into account the possibility of disasters that may affect multiple edges, nodes and data centers. Our objective was to minimize the network resources, in terms of wavelength links, used to han-

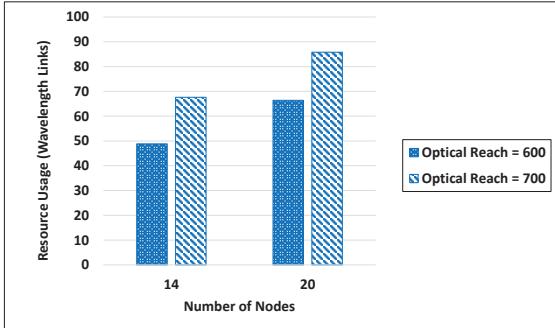


Figure 3: Variation of resource utilization with Optical Reach.

dle requests that will use backup paths in case of disasters. A novel feature of this approach is that we allow the possibility of multiple backup paths to handle different disasters for the same request for communication. Fiber channels for backup lightpaths are shared resources if they are used to handle different disasters, giving a new avenue for significant resource savings. Our algorithm was formulated as an ILP with the objective of reducing the resource usage and number of disasters that disrupt the primary lightpath. Our simulation results indicated that i) the resource usage increases with more disasters, ii) decreases with the number of replications of files and iii) lower optical reach leads to better resource usage, in terms of the number of wavelength-links used.

**Acknowledgements** This research was supported by Natural Sciences and Engineering Research Council of Canada through Discovery grants to Arunita Jaekel and Subir Bandyopadhyay.

## 5. REFERENCES

- [1] S. Bandyopadhyay, *Dissemination of Information in Optical Networks:: From Technology to Algorithms*.

- Springer Science and Business Media, 2007.
- [2] H. Zang, C. Ou, and B. Mukherjee, "Path-protection routing and wavelength assignment (rwa) in wdm mesh networks under duct-layer constraints," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 2, pp. 248–258, 2003.
- [3] C. S. Ou, J. Zhang, H. Zang, L. H. Sahasrabuddhe, and B. Mukherjee, "New and improved approaches for shared-path protection in wdm mesh networks," *Journal of Lightwave Technology*, vol. 22, no. 5, p. 1223, 2004.
- [4] M. F. Habib, M. Tornatore, M. De Leenheer, F. Dikbiyik, and B. Mukherjee, "Design of disaster-resilient optical datacenter networks," *Lightwave Technology, Journal of*, vol. 30, no. 16, pp. 2563–2573, 2012.
- [5] S. Ferdousi, F. Dikbiyik, M. F. Habib, M. Tornatore, and B. Mukherjee, "Disaster-aware datacenter placement and dynamic content management in cloud networks," *Journal of Optical Communications and Networking*, vol. 7, no. 7, pp. 681–694, 2015.
- [6] B. Mukherjee, M. Habib, and F. Dikbiyik, "Network adaptability from disaster disruptions and cascading failures," *Communications Magazine, IEEE*, vol. 52, no. 5, pp. 230–238, 2014.
- [7] P. K. Agarwal, A. Efrat, S. K. Ganjugunte, D. Hay, S. Sankararaman, and G. Zussman, "The resilience of wdm networks to probabilistic geographical failures," *IEEE/ACM Transactions on Networking (TON)*, vol. 21, no. 5, pp. 1525–1538, 2013.
- [8] S. Banerjee, S. Shirazipourazad, and A. Sen, "Design and analysis of networks with large components in presence of region-based faults," in *Communications (ICC), 2011 IEEE International Conference on*, pp. 1–6, IEEE, 2011.
- [9] [ftp://public.dhe.ibm.com/software/websphere/ilog/docs/optimization/cplex/ps\\_usrmancplex.pdf](ftp://public.dhe.ibm.com/software/websphere/ilog/docs/optimization/cplex/ps_usrmancplex.pdf) [retrieved on date: 26-oct-15], 2015.

Table 1: Comparison of blocking probability and time for Phase II with different network topologies, different traffic loads in Phase I, and different locations of datacenters.

Network	Datacenter locations	Phase I			Phase II		
		Disasters category	No. of requests	RU (#WL links)	BP	RU (#WL links)	Time (sec.)
14 Nodes	DC=[5,9]	I	20	79	0.03	7.22	0.01
		II	20	79	0	7.32	1.95
	DC=[3,12]	I	20	73.6	0	6.18	0.01
			25	99.8	0.17	5.26	0.01
	DC=[0,5,9]	II	20	76.2	0	6.14	1.22
			25	103.2	0.23	5.03	2.01
20 Nodes	DC=[1,7,18]	I	35	102.2	0.02	5.33	0.02
		II	35	101.4	0.03	5.72	1.6
		I	20	76.8	0	6.03	1.19
			25	97	0.10	5.59	1.02
			30	116.4	0.25	5.91	0.83
	DC=[1,7,10,18]	II	25	98.2	0.10	5.56	6.08
			20	63.6	0	5.12	1.62
		I	25	79.2	0.10	5	1.61
		II	20	66.6	0	5.01	6.54
			25	82	0.10	4.96	6.16

# A Resilient Self-Organizing Offshore Communication Network for Fishermen

Jennath HS, Nandesh Nair, Sethuraman Rao, Dhanesh Raj

Amrita Center for Wireless Networks and Applications (AmritaWNA)

Amrita School of Engineering

Amrita Vishwa Vidyapeetham, Amrita University

Kollam, Kerala ,India

jennath@gmail.com, {nandeshnair, sethuramanrao, dhaneshraj}@am.amrita.edu

Sujatha Narayanan

Extreme Networks Inc.

145, Rio Robles,

Jose, CA 95134, USA

sujatha@yahoo.com

Dilip Krishnaswamy

IBM Research

Bangalore, India

dilip@ieee.org

## ABSTRACT

Fishing is a major industry that provides the livelihood for millions of families worldwide. The fishing vessels are typically at sea for 5-7 days at a stretch completely cut off from the land. In a low-cost offshore communication system for fishermen currently being developed, a multilevel P2MP backhaul network connects clusters of boats at the fishing zones. Each cluster forms a wireless mesh network with multiple potential gateways to the backhaul network. This work proposes a resilient lightweight protocol for optimized dynamic provisioning of the backhaul network based on an adaptive, demand driven algorithm. A local controller, capable of running on any access router, selects the optimal subset of gateways so that the rest can be reconfigured as mobile base stations or switched to power saver mode. It assigns every boat in a cluster to the most appropriate gateway maximizing the utilization of gateways. A proof-of-concept implementation has been done with a hardware test bed and the results are very encouraging.

## CCS Concepts

- **Networks** → Network performance evaluation; Network services;

## Keywords

Backhaul network; Gateway; Controller; Long range Wi-Fi; Wireless Mesh network; Base station; P2MP; SDN

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

*ICDCN '17, January 04-07, 2017, Hyderabad, India*

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

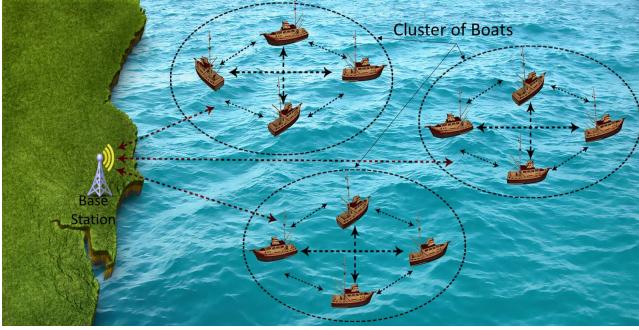
DOI: <http://dx.doi.org/10.1145/3007748.3018283>

## 1. INTRODUCTION

The main source of livelihood in several coastal villages across the world is small-scale fishing using traditional methods. A majority of these fishermen use boats that are poorly equipped in terms of security, communication and navigation and routinely spend 5-7 days at a stretch at sea completely cut off from the mainland. Thus, there is the need for an economical communication network which will enable fishermen to communicate with other vessels at sea as well as with the shore under both critical and non-critical circumstances. The communication systems currently employed by the fishermen are still the legacy VHF radios in broadcast mode, with line-of-sight coverage. In addition to this, fishermen use the cellular network for private calls; however, its coverage is limited to about 15 km from the shore. A low cost communication infrastructure employing a multilevel Point-to-Multi-Point (P2MP) architecture based on Long Range Wi-Fi (LR Wi-Fi) for backhaul and standard Wi-Fi mesh architecture for access is currently under development at our research center. The system architecture is described briefly in [1] [2]. The system consists of two tiers:

1. The long range Wi-Fi backhaul network is formed by onshore Base Stations (BS) and Customer Premises Equipment (CPE) deployed in the mid-sized boats (trawlers). CPE is the gateway to the backhaul network and can also play the role of a mobile base station to extend the range of the network. Therefore, it is referred to as Adaptive Backhaul Equipment (ABE).
2. The standard 802.11 Wi-Fi access network is formed by the interconnection of mesh enabled access points or Access Routers (AR) on board the boats within a fishing zone at sea. A fishing zone typically extends to a length of 4-8 km.

Figure 1 shows the network topology with three mesh network clusters. Such clusters get formed naturally when more than one boat reaches the same fishing zone. A cluster may have 4-5 mid-size boats (trawlers). If the fishing zone is closer to the shore, say within 20 km, the cluster may contain smaller boats also.



**Figure 1: Network Topology**

Each boat in a cluster is equipped with an AR and most trawlers will also have an ABE to provide the backhaul link to the base station at the shore. On these boats, the ABE is connected to the AR over Ethernet. For the sake of simplicity, only the first level P2MP backhaul network is shown in the Figure 1. One or more boats in a cluster can potentially provide a second (and subsequent) level P2MP network to extend the network range opportunistically. Boats are categorized into three different types based on the equipment they carry as shown in Table 1.

Equipment	Boat Size	Node Name
AR	Small	Access Node (AcN)
AR + ABE	Medium	Adaptive Node # (AdN)
AR + 2 * ABE	Large/Medium	Super Node (SuN)

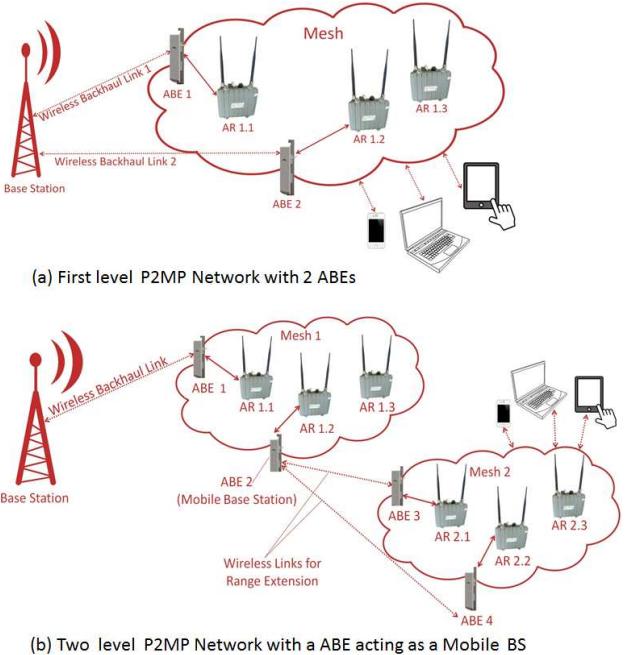
**Table 1: Types of Nodes**

# The node is adaptive because the ABE on board can be dynamically reconfigured as a mobile BS to opportunistically extend the range of the network.

LR Wi-Fi was chosen as the backhaul technology after a thorough comparative study of various options available [3]. Several field trials were conducted using both 2.4 GHz and 5 GHz equipment [1] [4]. A range of more than 45 km was realized in the first level shore-to-boat P2MP network and a range of more than 22 km was realized in the secondary boat-to-boat P2MP network. These sea trials were conducted from Azheekal beach in the state of Kerala in south west India.

In this marine communication system, a fishing zone with more than one fishing trawler frequently requires dynamic reconfiguration of the network for optimized traffic engineering due to the mobility of the boats. Hence there is scope for optimization in terms of maximum utilization of available capacity with a minimum number of active ABEs serving as gateways to connect (single-hop or multi-hop connectivity) with the base-station at the shore.

This work describes an adaptive and demand driven algorithm and a resilient light weight control protocol for optimal provisioning of the backhaul from the edge of the backhaul network (AR). The SDN paradigm of separating the control plane from the data plane is employed [5]. A local controller (LC), capable of running on any access router, selects the optimal subset of gateways so that the rest can be reconfigured as mobile BS. It assigns every mobile boat in a cluster



**Figure 2: Range Extension through Reconfiguration of ABE as a Mobile BS**

to the most appropriate gateway maximizing the utilization of gateways. It also elects a standby controller for resiliency.

Prior work in long range Wi-Fi networks [6] involved setting up a mesh network consisting of multiple point to point links. In this work, several point to multi-point long range Wi-Fi networks are stitched together over Ethernet and standard Wi-Fi mesh access networks. The number of long range Wi-Fi radio chains on a boat is limited to a maximum of two.

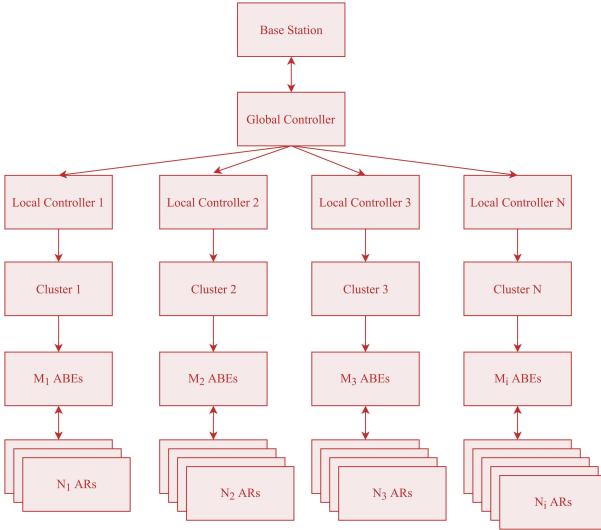
The remainder of the paper is organized as follows. Section 2 describes the system architecture. Section 3 provides the details of analysis and modelling for both mesh as well as backhaul links. Section 4 presents the communication protocol and the message exchanges in different scenarios. Section 5 presents the proposed algorithm. Section 6 details prototyping results and analysis. Finally, the paper is concluded in Section 7.

## 2. SYSTEM DESCRIPTION

Adaptive, demand-driven, energy-efficient provisioning of backhaul links for each boat in the cluster needs to be optimized dynamically from a local controller running on the AR on one of the boats in the cluster. Also, if this boat goes out of range, the controller should be migrated seamlessly to a different AR, i.e., the standby controller, in the cluster.

In this architecture, the traffic engineering control function is centralized at a local controller within each cluster and the location of the controller itself is software-defined dynamically by choosing a suitable AR in the cluster. Note that there will be one local controller per cluster, running at the P2MP network edge. A global controller (GC) will run on the onshore BS to manage the local controllers in the clusters (Figure 3). Control information updates will flow predominantly from the edge (AR) to the core (BS).

A cluster with  $N_i$  boats will have  $N_i$  ARs, one on each boat and  $M_i$  ABEs ( $M_i \leq N_i$ ), where the index  $i$  represents cluster  $i$ . A subset of boats will have an ABE connected to the AR

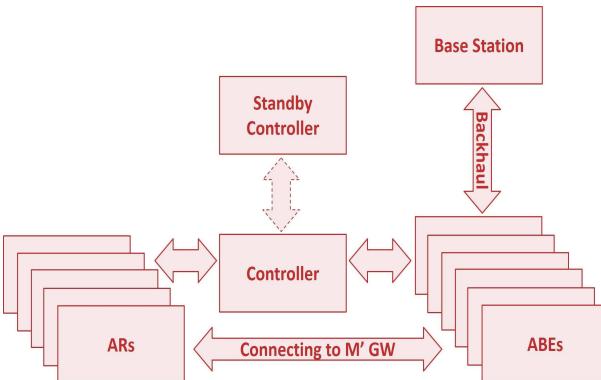


**Figure 3: System Architecture**

on the boat over Ethernet. ARs will form a mesh network connecting to each other, and use ABEs as the gateways to connect to the BS over the backhaul network. The number of ARs to be connected to an ABE will be limited by the ABE capacity and the anticipated traffic from the end user devices such as smart-phones, tablets, laptops, etc., connecting to each AR.

The controller controls the AR and ABEs in the cluster. The LR Wi-Fi communication network comprised of the BS and ABEs forms the backhaul network. The ARs deployed one on each boat in a cluster forms a mesh network and use the backhaul network to tunnel their traffic to the BS. The selection of ABEs to take part in the backhaul network is determined by the AR which has been elected as the “controller” by employing the proposed algorithm 4.

The controller evaluates the network elements and dynamically selects  $k_i$  out of  $M_i$  ABEs, where  $k_i \leq M_i$  to be part of the backhaul network. The controller also decides which ABE each AR should associate with as the gateway to the backhaul network. The proposed algorithm ensures selection of the best backhaul links and maximizes the utilization of the selected ABEs by minimizing the number of ABEs in use at any given time.



**Figure 4: Logical Network Diagram**

### 3. SYSTEM MODEL

The connections between ABEs and ARs could be viewed as an  $N_i \times M_i$  matrix ( $N_i$  rows for  $N_i$  ARs and  $M_i$  columns for  $M_i$  ABEs). At any point of time an AR will be connected with only one ABE. The matrix element is 1 if there is connection, otherwise it is 0. This implies,

$$\sum_{j=1}^{M_i} S_{mj} = 1 \quad (1)$$

where  $S_{mj}$  represents connectivity between the  $m^{th}$  AR and  $j^{th}$  ABE

Now, if all elements in a column are 0, that implies that the corresponding ABE is not being used at all. Our objective is to minimize the number of non-zero column vectors. At the same time, none of the ABEs should be oversubscribed. The traffic load on an ABE should be well within its capacity. The capacity can be computed based on Shannon's capacity theorem with a reduction factor introduced to account for practical real world scenario. In the proof-of-concept implementation, the ABE capacity is derived from the transmission rate recorded during the field trials for a given signal strength and noise.

For computing the utilization of an ABE, the traffic loads from all the ARs connected to it need to be estimated and added up. The downlink traffic is considered since it tends to be higher than the uplink traffic. This is done by determining the number of application type instances per AR (voice, video, data...) and then estimating traffic load per application type instance.

$$\text{Load}_{ARi} = a (\#\text{voiceapps})_{ARi} + b (\#\text{Tablets})_{ARi} + c (\#\text{Laptops})_{ARi} \quad (2)$$

Some ARs will connect to an ABE through an intermediate AR. In these cases, the best neighbour AR is chosen as the intermediate node based on the OLSR routing protocol metrics [7], Expected Transmission Time (ETT) and Expected Transmission Count (ETX). They represent the average time taken to transmit a packet successfully and the average number of transmissions per successful transmission respectively.

$$ETT = ETX \left( \frac{S}{BW} \right) \quad (3)$$

$$ETX = \frac{1}{d_f d_r} \quad (4)$$

$S$  : Size of the probe packet

$BW$  : Bandwidth of the link.(Data Rate)

$d_f, d_r$ : Packet delivery ratio in the forward and reverse directions.

### 4. COMMUNICATION PROTOCOL

The proposed light-weight control protocol runs periodically. The control message overhead during each schedule interval is reduced by having a centralized local controller. Based on the periodic updates from the participating ARs, LC decides the optimal network topology, i.e., which ABE each AR should be connected to. A secondary LC is used for fault tolerance (described in Section 5). Note that the algorithm used by the controller to assign ABEs to ARs is stateless. Therefore, there is no need for any state synchronization between the LC and the SLC.

## 4.1 Network Entities and Related Messages

The following are the network entities that use various message types for exchanging control information.

- Local Controller (LC): This is an AR elected as the LC and is responsible for the following:
  1. For broadcasting the Periodic Beacon Message (PBM) during each schedule to advertise the network and the controller information.
  2. For sending Controller Joining Response (CJR) in response to a Router Joining Request (RJR) from an AR.
- Standby Local Controller (SLC): For fault tolerance and redundancy, an SLC will be elected. This SLC will inherit control when the LC moves out of the network or becomes unresponsive.
- Access Router (AR): ARs form a mesh network. ARs connected with ABE initiate Controller Election Initiation Message (CEIM) when no beacon message is heard. All ARs except the initiator will respond to the election message with Election Participation Message (EPM). Also, in response to the Periodic Beacon Message (PBM) from LC, the new ARs in the network will send Router Joining Request (RJR) while the already connected ARs will send Periodic AR Update Message (PAUM). The parameters passed in these update messages are described in the optimization algorithm in Section 5.
- Adaptive Backhaul Equipment (ABE): ABE will not directly take part in the message exchange process. The AR connected to the ABE shares the ABE stats with the LC. The ABE will also relay the cluster topology information from LC to the global controller (GC) at BS.

## 4.2 Network Provisioning

Network provisioning is done dynamically based on a periodic schedule. The following are the steps involved in network provisioning.

1. ABEs will send the backhaul link quality information such as Received Signal Strength Indicator (RSSI), transmission rate, etc., to LC through the AR they are connected to, on a regular basis. AR will relay this information to LC along with its ID, ABE ID, cluster ID, etc.
2. ARs will send details of the anticipated end user traffic load to LC. They will also send the link quality info of their neighbour ARs to LC.
3. LC will grade the ABEs based on their capacity and SNR. LC filters the eligible ABEs based on a minimum threshold for SNR and link quality and sorts them in descending order based on their link quality. LC will walk through the list of ARs and connect each AR to the best possible ABE based on the ABE capacity and link quality, the AR load and the AR link quality (ETX) if there is an intermediate hop involved. The number of ARs to be connected to an ABE will be limited by the ABE capacity. Only the top  $n$  ABEs in the sorted ABE list will be enabled since they will have ARs connected to them and the rest can either be repurposed as mobile base stations to extend the network range or set to power saver mode.
4. If the link quality of the ABE has deteriorated or the

load on the ABE is beyond the capacity, LC will reassign the ARs connected to it to better ABEs. New ARs which are not yet part of the network will also be included in the evaluation to associate with an ABE.

## 5. OPTIMIZATION ALGORITHM

The Network Resource Optimization (NRO) algorithm run by the LC for selecting the optimal gateway for each AR is described below. The main difference from prior work in this area [8] [9] [10] [11] is that the gateway assignment is made centrally in a local controller in this case as opposed to being made by the individual nodes themselves. This enables the introduction of additional traffic engineering and load balancing in a greedy allocation algorithm by considering the capacity and load of each gateway and offloading some traffic to sub-optimal gateways as needed.

The controller software module will be installed on all ARs. However, the election algorithm as described in Section 4 will elect LC and SLC from among the ARs in the cluster. ARs send the SNR and link quality of the ABE

---

1: **procedure** 1: PERIODICUPDATEFROMAR  
 2:     ▷ *AccessRouters(ARs)* communicate the following information to controller periodically every minute.

- *SNR, ID info, ABE ID info, Cluster ID info* of directly connected ABE. ABE every  $x$  minutes to the controller.
- SEND stats of neighbor ARs to the controller.
- SEND the number and type (smartphone, tablet, laptop, etc.) of end user nodes connected. (This will also serve as a keepalive message.)

▷ Perform the steps 2 through 6 after each keepalive.  
 3: **end procedure**

---



---

**procedure** 2: FILTERANDRANKABES  
 2:     ▷ FILTER the eligible ABEs based on a minimum threshold for SNR  
 4:       **for** (EACH *abe* in *globalAbeList*) **do**  
 5:           **if** (*abe.linkSNR*  $\leq$  *MIN\_SNR\_THLD*) **then**  
 6:              *globalQualAbeList.add(abe)*  
 7:           **end if**  
 8:       **end for**  
 8:       ▷ Rank ABEs based on their SNR values  
 9:       *globalAbeRankedList* =  
 10:       *globalQualAbeList.rankBySnr()*  
 10:     ▷ Let the Normal Load Limit for a abe be *NLL* and the Exception Load Limit for a abe be *ELL*  
 11:       ▷ COMPUTE Load Limits based on capacity  
 12:        ▷ *ELL\_factor* = 0.95  
 12:        ▷ *NLL\_factor* = 0.80  
 14:       **for** (EACH *abe* in *globalAbeRankedList*) **do**  
 15:           *abe.NLL* = *abe.capacity* \* *NLL\_FACTOR*  
 16:           *abe.ELL* = *abe.capacity* \* *ELL\_FACTOR*  
 18:       **end for**  
 19: **end procedure**

---

they are connected to, their ID, ABE ID, etc., to LC periodically. They also provide a report of their neighbour ARs.

---

```

procedure 3: COMPUTEARLOADSANDRANKNEIGH-
BOURS
3:   for (EACH ar in globalArList) do
      ar.rankedNeighborList =  

      ar.neighborList.rankByEttx()  

      ar.computeLoad();
6:   end for
end procedure



---


procedure 4: ASSIGNING ABEs TO ARs
  ▷ Iterate through the globalAbeRankedList
  for (EACH abe in globalAbeRankedList) do
4:    abe.isAssigned = FALSE
    if ((abe.directlyConnAr.abe == NULL) AND  

    (abe.load+abe.directlyConnAr.load) ≤ abe.NLL) then
      abe.directlyConnAr.abe = abe
      abe.assignedArList.add(abe.directlyConnAr)
8:    abe.isAssigned = TRUE
    end if
    for (EACH neighborAr in  

abe.directlyConnAr.rankedNeighborList) do
      if (neighbor.ar == NULL AND  

      neighbor.ETX ≤ ETX_THLD AND  $\leftarrow$  (abe.load +  

      neighbor.load) ≤ abe.NLL) then
        neighbor.ar = abe
        abe.assignedArList.add(neighbor)
        abe.isAssigned = TRUE
      end if
    end for
  end for
end procedure



---


procedure 5: EXCEPTION HANDLING
  ▷ If there is enough ABE overcapacity  

  in the network, all ARs must have been assigned to an  

  ABE by now. Handle the exception cases below
  ▷ Loop through the list of ARs and try to assign the  

  unassigned ARs to a ABE

5:   for (EACH ar in globalArList) do
    if (ar.abe == NULL) then
      for (EACH neighbor in  

ar.rankedNeighborList) do
        if (neighbor.directlyConnAbe.  

load+ar.load) ≤ (neighbor.directlyConnAbe) then
          ar.abe = neighbor.directlyConnAbe
          neighbor.directlyConnAbe.  

assignedArList.add(ar)
          neighbor.directlyConnAbe.
          isAssigned = TRUE
        end if
      end for
    end if
  end for
15: end for
  ▷ if ar.abe == NULL at this point, the AR will remain  

  without an ABE assigned until the next keepalive
end procedure

```

---

The OLSR metric, Expected Transmission Count (ETX) [9] [12] [13], is used to provide the neighbour AR link quality

---

#### **procedure** 6: PROVISIONING THE ABE

▷ The controller communicates with the ARs to whom the candidate ABEs are connected to turn them on or off respectively as needed

---

**end procedure**

---

information. ARs will also send the estimated traffic load of the end user devices connected to them to LC during each schedule.

The assumption is that we have an overcapacity of ABEs in the cluster such that some ABEs can be selected as gateways, whereas the remaining ABEs can be repurposed as mobile BS or disabled to save energy in the system. This algorithm dynamically identifies the best subset of ABEs that can be selected as gateways to allow the remaining ABEs to function either as mobile BS or to be switched to power saver mode. Note: Link quality of ABE is determined by the parameters, data rate and SNR, whereas the link quality of AR is determined by the OLSR parameter ETX.

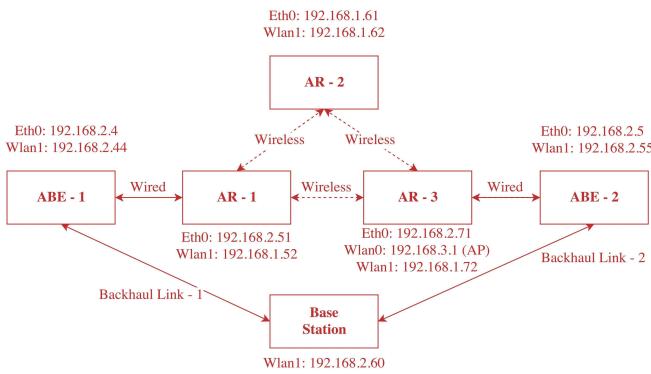
## 6. PROTOTYPE IMPLEMENTATION

For a proof of concept implementation of the controller algorithm and the communication protocol, a hardware prototype test bed consisting of three ARs, two NanoStation M2 ABEs and a Rocket M2 base station from Ubiquiti Networks is set up. The ARs are built in-house using Alix 3d3 based embedded system board and Linux Voyage OS. The goal of this prototyping exercise is to demonstrate the need for periodic dynamic reconfiguration of the network in order to optimize the utilization of the ABEs and maximize network range.

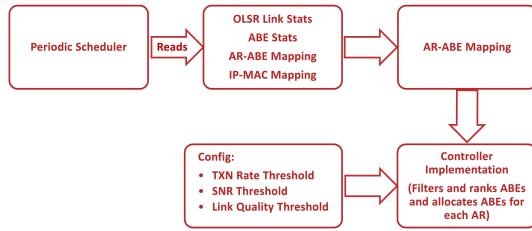
Two out of the three ARs are connected over Ethernet to an ABE each. These two ABEs connect to the BS using LR Wi-Fi and provide backhaul to the mesh network formed by the three ARs (Figure 5). The algorithm described in Section 5 is used for provisioning the backhaul for the communication network. The controller implementation is done using Java Spring Framework [14]. One of the ARs with ABE, statically chosen as the LC, is deployed with the custom controller implementation.

The periodic update interval from the ARs to the LC is set as 60 seconds. The trawler boats travel at a speed of 6-7 kmph while fishing in the fishing zone. Assuming a speed of 6 kmph, the boat would have travelled 100 m in 60 sec. Therefore, re-evaluating the connectivity within the mesh network and in the backhaul network every 60 sec seems to be a reasonable choice. The interval should not also be too short so as not to overwhelm the LC with too much computational load.

OLSR daemons [15] [16] [17] are deployed on all the three ARs and initiated at start-up to periodically collect stats such as link quality, neighbour link quality, ETX, etc.[9]. The ABEs from Ubiquiti Networks have air OS installed on them and provide interfaces which can be invoked by Shell Scripts at regular intervals to fetch the backhaul stats such as received signal strength, noise floor, transmission rate, etc. The controller checks the network stats at frequent intervals and executes the backhaul selection algorithm to decide the network topology. Figure 6. depicts the implementation as a block diagram.



**Figure 5: Hardware Test Bed Setup**



**Figure 6: Implementation Block Diagram**

## 6.1 Test Scenarios

Four different test scenarios are exercised as shown in Table 2. In all cases, the ABE with better link quality is provisioned. An unassigned ABE can go to power saver mode to save energy in the system, or it can be converted into a mobile base-station to further extend the range of the network if needed. Thereby, the spare ABE can be reconfigured as a mobile BS to extend the network range or simply switch to power save mode.

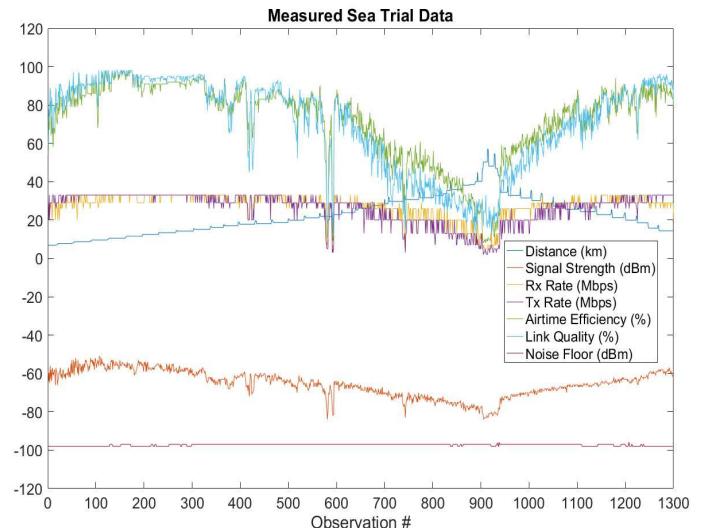
When the SNR falls below 15 dB, the link quality is considered to be poor. During the field trials conducted over the Arabian Sea, the noise floor was observed to be -98 dBm. Therefore, this translates to an RSSI threshold of -83 dBm for link quality. If RSSI falls below this threshold, the link quality is poor.

Test Case	ABE1 Stats	ABE2 Stats	Resulting Topology	Unassigned ABE
1	RSSI = -58 Txn Rate = 33	RSSI = -60 Txn Rate = 29	AR1,AR2,AR3,AR4 = ABE1	ABE2*
2	RSSI = -55 Txn Rate = 29	RSSI = -52 Txn Rate = 33	AR1,AR2,AR3,AR4 = ABE2	ABE1*
3	RSSI = -80 Txn Rate = 5	RSSI = -76 Txn Rate = 20	AR1,AR2,AR3,AR4 = ABE2	ABE1*
4	RSSI = -69 Txn Rate = 33	RSSI = -77 Txn Rate = 10	AR1,AR2,AR3,AR4 = ABE1	ABE2*

\* The unassigned ABE can be reconfigured as a Mobile BS or can be switched to power saver mode.

**Table 2: Resultant Network Topology under Various Scenarios**

To summarize, an ABE that has a poorer link quality



**Figure 7: Sea Trial Measured Data**

is kept unassigned (ABE1 is unassigned in Test Case 3 and ABE2 is unassigned in Test Case 4). When both ABEs have good link quality, the the ABE with the worser link may be unassigned (Test cases 1 and 2). An unassigned ABE can go to power saver mode to further extend the range of network if needed.

Figure 7 illustrates the link quality data captured during the sea trial. In general the RSSI value displays a direct measure of link quality, which indicates monotonous relation with the transmission rate. However the actual transmission or reception rate depends on other factors such as packet delivery ratio, fading effects, ducting effects etc, over the sea surface. This explains the variations in transmission rates observed at nearly same RSSI levels.

## 7. CONCLUSION

This work involves design and implementation of a novel adaptive, demand aware, energy efficient algorithm and protocol for provisioning the backhaul links of a marine offshore communication network. The controller, one per cluster of boats, is run at the edge (AR) of the backhaul network. The algorithm covers the election of controller, and initial and continuous periodic provisioning of the backhaul links. The algorithm is implemented in a custom designed controller developed using Java Spring Framework. The communication network is modelled using real hardware. A small-scale prototype developed using a combination of physical and simulated devices in this work yielded encouraging positive results.

## 8. ACKNOWLEDGEMENT

We are immensely grateful to our beloved Chancellor, Dr. Mata Amritanandamayi Devi, for the inspiration and motivation provided by her. This project is partly funded by a grant from Information Technology Research Agency (ITRA), Ministry of Electronics & Information Technology (MeitY), Govt. of India. The Access Router was developed by our partner institution, IIST, Trivandrum.

## 9. REFERENCES

- [1] S. N. Rao, D. Raj, S. Aiswarya, and S. Unni. Realizing cost-effective marine internet for fishermen. In *2016 14th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pages 1–5, May 2016.
- [2] S. Rao, M. V. Ramesh, and V. Rangan. Mobile infrastructure for coastal region offshore communications and networks. In *IEEE Global Humanitarian Technology Conference*, pages 69–83, USA, 2016. IEEE.
- [3] H. S Jennath, M. K Anju, D. Raj, and S. Rao. Comparative study of backhaul options for communication at sea. In Dr. Rajesh K Bhatia and Dr. Janahanlal Stephen, editors, *Sixth International Conference on Recent Trends in Information, Telecommunication and Computing*, pages 69–83, Chennai, India, 2015. McGraw-Hill.
- [4] S. Unni, D. Raj, K. Sasidhar, and S. Rao. Performance measurement and analysis of long range wi-fi network for over-the-sea communication. In *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), 2015 13th International Symposium on*, pages 36–41, May 2015.
- [5] A. Detti, C. Pisa, S. Salsano, and N. Belfari-Melazzi. Wireless mesh software defined networks (wmsdn). In *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 89–95, Oct 2013.
- [6] Rabin Patra, Sergiu Nedevschi, Sonesh Surana, Anmol Sheth, Lakshminarayanan Subramanian, and Eric Brewer. Wildnet: Design and implementation of high performance wifi based long distance networks. In *Proceedings of the 4th USENIX Conference on Networked Systems Design & Implementation*, NSDI'07, pages 7–7, Berkeley, CA, USA, 2007. USENIX Association.
- [7] Andreas Tonnesen, Andreas Hafslund, and Oivind Kure. The unik olsr plugin library. In *OLSR Interop and Workshop*, August 2004.
- [8] F. P. Setiawan, S. H. Bouk, and I. Sasase. An optimum multiple metrics gateway selection mechanism in manet and infrastructured networks integration. In *2008 IEEE Wireless Communications and Networking Conference*, pages 2229–2234, March 2008.
- [9] U. Ashraf, S. Abdellatif, and G. Juanole. Gateway selection in backbone wireless mesh networks. In *2009 IEEE Wireless Communications and Networking Conference*, pages 1–6, April 2009.
- [10] Takeshi Matsuda, Hidehisa Nakayama, Xuemin (Sherman) Shen, Yoshiaki Nemoto, and Nei Kato. Gateway selection protocol in hybrid manet using dymo routing. *Mob. Netw. Appl.*, 15(2):205–215, April 2010.
- [11] Yi Fu, Kwang-Mien Chan, Kean-Soon Tan, and Boon-Sain Yeo. Multi-metric gateway discovery for manet. In *2006 IEEE 63rd Vehicular Technology Conference*, volume 2, pages 777–781, May 2006.
- [12] P. M. Esposito, M. E. M. Campista, I. M. Moraes, L. H. M. K. Costa, O. C. M. B. Duarte, and M. G. Rubinstein. Implementing the expected transmission time metric for olsr wireless mesh networks. In *2008 1st IFIP Wireless Days*, pages 1–5, Nov 2008.
- [13] M. E. M. Campista, P. M. Esposito, I. M. Moraes, L. H. M. k. Costa, O. C. M. b. Duarte, D. G. Passos, C. V. N. De Albuquerque, D. C. M. Saade, and M. G. Rubinstein. Routing metrics and protocols for wireless mesh networks. *IEEE Network*, 22(1):6–12, Jan 2008.
- [14] Stefano Salsano, Giuseppe Siracusano, Andrea Detti, Claudio Pisa, Pier Luigi Ventre, and Nicola Belfari-Melazzi. Controller selection in a wireless mesh sdn under network partitioning and merging scenarios. *arXiv preprint arXiv:1406.2470*, 2014.
- [15] Thomas Clausen, Philippe Jacquet, Cédric Adjih, Anis Laouiti, Pascale Minet, Paul Muhlethaler, Amir Qayyum, and Laurent Viennot. Optimized Link State Routing Protocol (OLSR), 2003. Network Working Group.
- [16] Thomas Heide Clausen and Philippe Jacquet. Optimized link state routing protocol (olsr). Experimental RFC 3626, IETF, Project Hipercom, INRIA, October 2003.
- [17] Dilpreet Kaur and Naresh Kumar. Comparative analysis of aodv, olsr, tora, dsr and dsdv routing protocols in mobile ad-hoc networks. *International Journal of Computer Network and Information Security*, 5(3):39, 2013.

# Performance Assessment of an Extremely Challenged Mobile Infrastructure Network over the Oceans

Dhanesh Raj, Vickram Parthasarathy, Sethuraman N Rao, Maneesha V Ramesh

Amrita Center for Wireless Networks and Applications (AmritaWNA)

Amrita School of Engineering

Amrita Vishwa Vidyapeetham, Amrita University

Kollam, Kerala ,India

{dhaneshraj, vickramp, sethuramanrao, maneesha}@am.amrita.edu

## ABSTRACT

A novel cost-effective network architecture for providing Internet connectivity to marine fishermen has been successfully prototyped by our research center. A pilot deployment is in progress in a coastal Indian village. This will improve the quality of life of the financially constrained marine fishermen who spend 5-7 days offshore on average for a single fishing trip; it will also help in their safety and security. The architecture employs multiple long range Wi-Fi (LR Wi-Fi) based infrastructure networks stitched together as backhaul. The access network consists of Ethernet and Wi-Fi mesh. The fishermen connect to the on board Wi-Fi access point cum router using their smart phones and are able to use all the apps and services on their smart phone. While the primary infrastructure network uses onshore base stations, the secondary infrastructure networks use boats as mobile base stations. Three field trials were conducted over the ocean using one onshore base station and two mid-sized boats known as trawlers. The performance of both primary and secondary infrastructure networks was assessed during these field trials. This paper describes the impressive results obtained in assessing the performance of the secondary infrastructure network.

## CCS Concepts

•**Networks** → Network experimentation; Network measurement; *Network performance analysis*;

## Keywords

Adaptive Long range Wi-Fi Network; Opportunistic Mobile Infrastructure Network; Point to Multi-Point Network; Smart phone, Marine Internet

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](http://permissions.acm.org).

ICDCN '17, January 04-07, 2017, Hyderabad, India

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3018284>

## 1. INTRODUCTION

Marine fishing has been the traditional occupation of fishing communities living along the coasts in many countries of the world for several centuries. India is one such country with a coastline stretching over more than 5000 miles with thousands of fishing villages dotting the coastline [1] [2]. The fishing community consists of boat owners and fishermen who are employed by the boat owners. Some people go fishing in their own boats playing both the roles. Such people tend to own the smaller boats and their fishing trips typically last for a day or two going not too far from the shore (15-20 km). The mid-sized boats known as trawlers have the capacity to store much more fish in onboard cold storage. Up to 10 fishermen go out on a trawler for fishing trips lasting 5-7 days on average. They use special nets and techniques depending on the type of fish they are targeting and could go as far as 120 km away from the shore in search of fish. The fishermen tend to be economically underprivileged especially in developing countries. Currently, they don't have a cost-effective way to communicate with the shore while on a fishing trip. Satellite communication is expensive and not affordable for their income levels. The trawlers are fitted with walkie-talkies for boat to boat communication; however, their range is short and they are not reliable under adverse weather conditions.

Two years ago, our researchers interviewed nearly a hundred of these fishermen in two villages to gain in-depth understanding of the problems faced by them, of the type of equipment that is currently owned by them and of their requirements and skill levels. The interviews revealed that about 67% of fishermen owned smart phones. This number is on the rise with the falling prices and growing popularity of smart phones. Prolonged isolation from their families was felt as a real problem by the fishermen with no viable solution in sight. Most importantly collision with a ship at night was cited as the biggest potential hazard by the fishermen [3]. The fishermen were tech-savvy in general and were willing to try out new solutions, if presented. The interviews also brought out a clear understanding of their fishing behavior. They tend to form clusters of boats in the fishing zones once the zones are located using a combination of their intuition and technology. The trawlers are fitted with echo sounders for under-sea viewing.

Having understood the needs and the behavioral traits of fishermen, our researchers set out to find a suitable solution that would fit their profile. The goal was to use an access network that would enable the use of smart phones already

owned by the fishermen and build a backhaul network using the cheapest option available. This approach would ensure that both the capital expenses and the operating expenses are kept low. Several technology options for backhaul such as 2G/3G, LTE, CR, WiMAX and Wi-Fi were evaluated and long range Wi-Fi (LR Wi-Fi) was chosen as the technology of choice for backhaul [4]. The inference was based on a comparative evaluation of several parameters such as (capital and operating) cost, range, availability, data rates, etc. The price of Wi-Fi equipment has been declining especially with the introduction of the next generation 802.11ac technology. Several vendors in the market supplying LR Wi-Fi gear and standard access points (AP) make the technology easily available. The fact that the technology uses unlicensed spectrum helps to keep the costs lower.

A novel and unique network architecture for the backhaul network is employed to serve the complex needs of providing connectivity to hundreds of boats as far away as 120 km from the shore in an extremely dynamic and mobile environment. The architecture employs multiple long-range Wi-Fi (LR Wi-Fi) based infrastructure networks stitched together opportunistically as backhaul. The access network consists of Ethernet and Wi-Fi mesh.

The fishermen connect to the on board Wi-Fi access point cum router using their smart phones and are able to use all the apps and services on their smart phone. While the primary infrastructure network has an onshore base station with Internet uplink, the secondary infrastructure networks use boats as mobile base stations. The architecture of the system, named OceanNet, is described in section 2.

Prior work using LR Wi-Fi [5] [6] was done using a mesh network with multiple point-to-point links primarily on land. Our architecture uses point-to-multi-point links as in an infrastructure network. Therefore, while some of the problems that were tackled in these projects such as interference are not so relevant to us, our architecture presents its own challenges due to the complexities of the problem domain such as extreme mobility, instability of the sea surface, very long range, etc.

Triton is a project that attempts to provide high speed and low cost maritime communication [7]. It uses 802.16 wireless mesh nodes equipped with directional antenna arrays. Two hop measurements were done at 5.8 GHz. The range obtained over the first hop (first boat to onshore base station) was 1.3 km and over the second hop (second boat to first boat) was 1 km.

Marcom proposed a Wireless Coastal Area Network (WiCAN) architecture using sub-GHz WiMAX for backhaul and Mobile Multi-hop Relay (MMR) and wireless mesh for range extension [8]. However WiMAX has been superseded by LTE-Advanced technology.

Bluecomplus aims to come up with a proof of concept for a communication solution enabling cost effective broadband internet access at remote ocean areas [9]. The proposed solution is based on standard Wi-Fi access. Several intermediate helikites and ocean platforms are used as relay nodes from the onshore base station into the ocean.

Satellite communication is used only by luxury cruisers and yachts. There are companies like Speedcast [10] and Axxess Marine [11] which offer unlimited internet usage for superyachts. This solution is expensive and therefore not affordable for the fishermen.

In order to validate the assumptions made in the design of

our architecture, three field trials were conducted over the Arabian Sea from a coastal village in the state of Kerala in south west India. Both the primary and secondary infrastructure networks were evaluated. The results of the evaluation of primary infrastructure network have been published already [12] [13]. The secondary infrastructure network was evaluated extensively during the third field trial. Section 3 on the validation of the proposed architecture describes the results of the evaluation of secondary infrastructure network in detail. The concluding remarks and future action plan are presented in section 4 .

## 2. OCEANNET ARCHITECTURE

As mentioned before, OceanNet's backhaul network is formed by opportunistic stitching together of multiple LR Wi-Fi infrastructure networks. Figure 1 depicts the OceanNet architecture. Each of these infrastructure networks is a point-to-multi-point (P2MP) network wherein multiple client stations connect to a base station. The primary P2MP network has its base station on the shore at a height of 50-60 m from the ground. The onshore base station is connected to the Internet. The client stations, aka Customer Premises Equipment (CPE), are mounted on a pole on the boat at a height of about 8-10 m from the water level. The CPE on the boat is also capable of acting as a base station and can change its role dynamically depending on the location of the boat and the traffic characteristics of the network. For this reason, we refer to it as Adaptive Backhaul Equipment (ABE). The ABE, acting as a mobile base station, can form a secondary P2MP network.

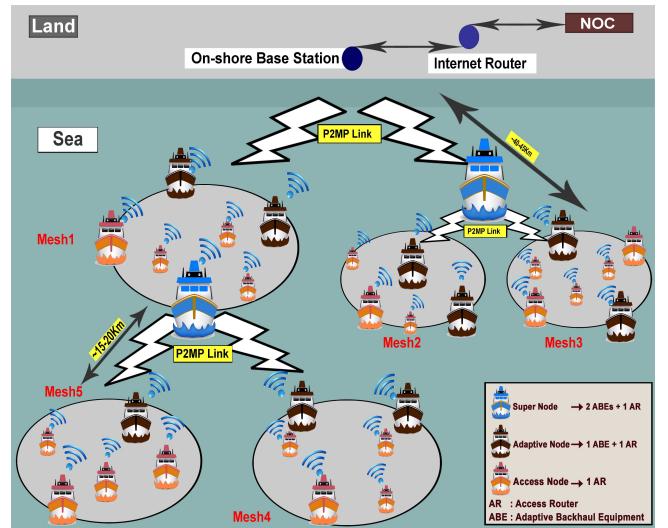


Figure 1: OceanNet Architecture

The ABE is connected over Ethernet to a standard Wi-Fi access point (AP) cum router collocated on the boat. We refer to it as Access Router (AR). Fishermen's smart phones, tablets, etc., connect to the AR using their Wi-Fi radio. This enables the end user devices to get internet access and thereby the fishermen can use all the apps and services that they have installed on their smart phones or tablets during their fishing trips.

Boat clusters get formed organically in the fishing zones, sometimes helped by the word spreading within the buddy

network of fishermen over walkie-talkies once a fishing zone is detected. Such clusters could be heterogeneous consisting of small boats and mid-sized boats (trawlers). In some cases, a trawler transports one or two small boats to and from the fishing zone along with it. Two small boats can be attached to the sides of a trawler. The small boats will typically have only an AR on board. The trawlers will typically have an AR and one or two ABEs on board.

The ARs within a cluster can form a wireless mesh network for access. This way, the AR on a small boat (or a trawler) can connect to the backhaul network through an ABE on a neighboring trawler. Depending on the traffic needs of the network and the relative positions of the boats, a subset of ABEs within a cluster will act as gateways to the backhaul network. Another subset of ABEs will act as mobile base stations. Some may be turned off or switched to power saver mode, wherein only the receiver may be turned on. A controller running on an AR within the cluster will provide traffic engineering services determining the optimal role of each ABE within a cluster periodically. It will also determine the optimal gateway for each AR in the cluster periodically and notify the ARs of any changes.

The SDN paradigm of separating the control plane from the forwarding plane is applied and the control plane functionality is encapsulated within the controller. The controller is stateless and therefore can relocate itself to a different AR fairly quickly when the need arises due to the extreme mobility of the nodes in the network. This is done by designating a standby controller in each cluster. When the active controller is not reachable, the standby controller takes over as the active controller and designates a new standby controller. When both the active and standby controllers have moved out of range, a new active controller is elected by the cluster members. The new active controller in turn designates a new standby controller.

The onshore Network Operations Center (NOC) is used to monitor and manage the network as well as to provide value-added applications and services such as an early warning system for collision alerts, border crossing alerts, etc., location tracking of fishing vessels and so on [14].

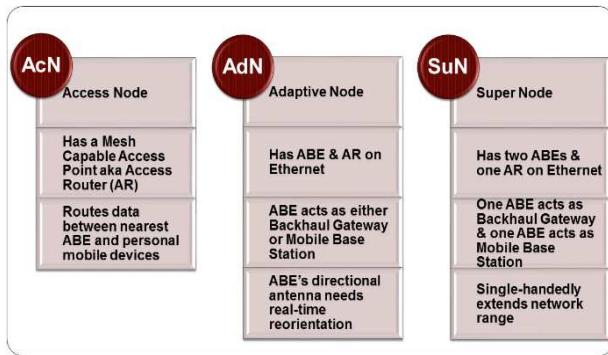


Figure 2: Node Types - AcN, AdN and SuN

Note that a trawler with two ABEs and an AR on board can be configured to have one of the ABEs acting as a gateway to the backhaul network and the other ABE as a mobile base station. This way, it can single-handedly extend the range of the network. Such a boat is designated as a Super Node (SuN). A trawler with one ABE and an AR is desig-

nated as an Adaptive Node (AdN). The ABE on an AdN can dynamically switch its role between being a gateway to the backhaul network and being a mobile base station. It could also operate in power saver mode if needed. A mobile base station will operate in power saver mode when it has no client stations connected to it. In this mode, it will send out beacons periodically and listen for any incoming connection requests. A boat with only an AR on board is designated as an Access Node (AcN). Figure 2 describes the node types, AcN, AdN and SuN.

Note that the primary infrastructure network has a higher range because the onshore base station can be mounted at a much higher elevation. Also, the coverage within the primary infrastructure network is guaranteed since the onshore base station is fixed and always available. The secondary infrastructure networks, however, are formed opportunistically based on the presence of boats within range to be used as mobile base stations. Due to this reason, there could be temporary loss of connectivity for some boats and the network should be delay tolerant. The nodes should be able to buffer messages until a suitable relay node is found towards the destination.

### 3. VALIDATION OF THE OCEANNET SOLUTION ARCHITECTURE

Three field trials over the Arabian Sea have been conducted from a coastal village in Kerala, India to validate the above network architecture for providing Internet connectivity to marine fishermen. FCC compliant LR Wi-Fi equipment from Ubiquiti Networks was used in the field trials. Rocket M base stations [15] were used onshore and Nano-stations [16] were used as ABEs on boats. The field trial setup is shown in Figure 3.



Figure 3: Field Trial Setup

#### 3.1 Field Trial 1

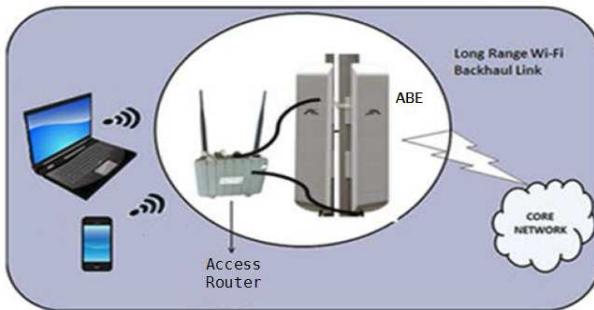
The first field trial was conducted using one trawler equipped with a 5 GHz LR Wi-Fi ABE and an AR along with an onshore base station at a height of 56 m. The ABE was mounted on the boat at a height of 9 m from the sea level. The channel bandwidth was set to 5 MHz. The maximum range obtained was only 17 km using the primary infrastructure network. Two factors contributed towards the poor results: (i) The ABE setup was imperfect and was shaking

vigorously in the winds. This impacted antenna alignment. (ii) The noise level recorded by the ABE was as high as -83 dBm. In the subsequent trials, it was -98 dBm or lower. The range of 17 km is only slightly better than the 12-15 km range obtained using the onshore cellular network. The results are presented in [12].

### 3.2 Field Trial 2

In the second field trial, 2.4 GHz LR Wi-Fi ABE was used. A trawler fitted with an ABE and an AR was taken to the sea to measure the maximum range from the onshore base station at a height of 56 m (as in the first field trial). The results were far more impressive. Internet connectivity was established as far as 45.6 km away from the shore. Mobile apps such as Whatsapp and Skype were run successfully. The range of the secondary infrastructure network was also measured and found to be 18 km. The second trawler used as the base station of the secondary infrastructure network was anchored on the shore; hence, this was only an approximate assessment of the secondary infrastructure network. The results were quite encouraging and we could expect to get better results over the ocean where there would be much less interference due to multi-path and due to external sources. The detailed analysis of the results of the second field trial is covered in [13].

### 3.3 Field Trial 3

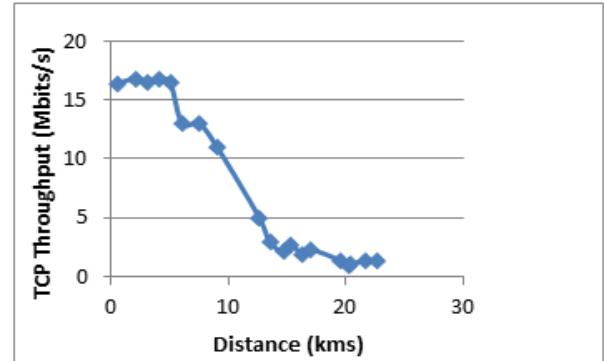


**Figure 4: The Access Network on board a Super Node**

During the third field trial, two trawlers were used. One trawler was a Super Node as it had two ABEs operating at 2.4 GHz with their backs turned towards each other. The two ABEs were facing in opposite directions. The other trawler had two ABEs, one operating at 2.4 GHZ and one at 5 GHz. Therefore, it was acting as two Adaptive Nodes, one at each frequency. Two onshore base stations were mounted side by side at a height of 56 m, one operating at 2.4 GHz and one operating at 5 GHz. Figure 4 shows the access network on board a Super Node. Note that the Adaptive Node will have a similar access network, the only difference being that there will be only one ABE (instead of two) connected over Ethernet to the access router, a range of 43.7 km was achieved in the 2.4 GHz primary infrastructure network. This was comparable with the results obtained in the second field trial using 2.4 GHz gear. The 5 GHz primary infrastructure network recorded a range of 41.1 km, much

better than the results obtained in the first field trial using 5 GHz equipment.

By using one of the ABEs on the SuN (trawler #1) as a mobile base station to extend the network range and getting the AdN (trawler #2) to connect to it, a secondary infrastructure network was formed. The maximum range achievable in the secondary infrastructure network was 22.6 km. TCP and UDP performance of the network was measured using iPerf. Even at the farthest distance within range, i.e., at a distance of 22.6 km, TCP throughput of nearly 1 Mbps was obtained.



**Figure 5: TCP Throughput versus Distance in the Secondary P2MP network**

Figure 5 depicts the variation of TCP throughput with distance in the secondary infrastructure network. The TCP throughput is > 16 Mbps up to a distance of 5 km. Then it falls steadily to 1 Mbps at a distance of 14 km. Thereafter, the fall is much more gradual and the throughput is about 0.75 Mbps even at a distance of 22.6 km. We measured a good RSSI at 22.2km for boat to boat communication as shown in Figure 6. Table 1 summarizes the configuration parameters and the results obtained during the three field trials.

#	Configuration	Results	Apps Run
1	Frequency: 5.8 GHz, 1 Boat (Adaptive Node), Base Station antenna height: 56 m above the ground, Boat antenna height: 9 m above sea level	Boat <=>Shore connectivity range: 17.7 km	Browser, ping
	Frequency: 2.4 GHz, 2 Boats (Adaptive Nodes), Same antenna heights as above	Boat <=>Shore connectivity range: 45.6 km, Boat <=>Boat connectivity range: 16 km	Browser, Whatsapp, Skype audio/video, YouTube, ping
3	Frequency: 2.4 GHz and 5.8 GHz, 2 Boats (1 Adaptive Node, 1 Super Node), Base Station antenna height: 56 m, Boat antenna height: 8m, 9 m	<b>2.4GHz:</b> Shore <=>Boat: 43.7 km, Boat <=>Boat: 22.6 km.  <b>5.8 GHz:</b> ,Shore<=>Boat: 41.1 km, TCP Throughput: 0.75 - 4 Mbps, UDP Packet Loss: 0 - 7 %	Browser, Whatsapp, Skype audio/video, ping, iPerf

**Table 1: Summary of Field Trials & Results**

Note that as more ABEs join the network, the throughput per ABE will be lower. This is because the ABEs use TDMA for medium access. The effect of more ABEs join-

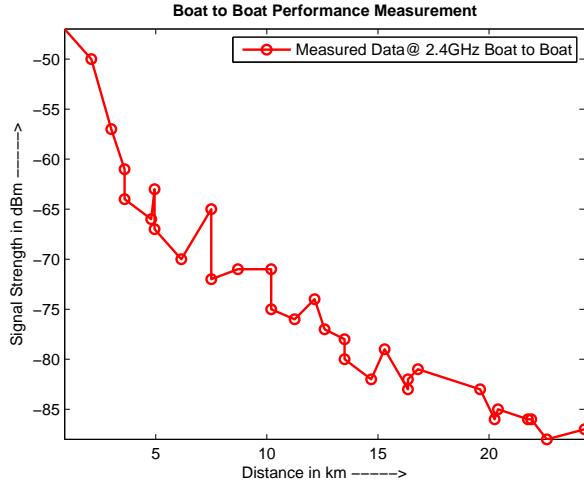


Figure 6: RSSI vs Distance between trawler#1 and trawler#2-Secondary P2MP network

ing the network on the TCP throughput was studied across the backwaters spanning 500 m. The same base station at a height of 56 m was turned towards the land and connectivity was established with ABEs mounted at about the same height of 9m (as on the boat) on the other shore of the backwaters. The results of this study are shown in Figure 7.

The throughput can be increased by increasing the channel bandwidth. By increasing the channel bandwidth to 20 MHz, we achieved an average TCP throughput of 10.3 Mbits/sec for the 10 ABEs scenario over the backwaters. During our sea trials, we observed that increasing the channel bandwidth results in a slight increase in the path loss thereby resulting in somewhat shorter range. This observation was also confirmed in our tests over the backwaters using the same LR Wi-Fi equipment. We observed a 10 dB drop in signal strength as the channel bandwidth was increased from 5 MHz to 40 MHz. We also observed that the noise floor value went higher as we increased the channel bandwidth. The noise floor value went up from -98 dBm to -89 dBm as the channel bandwidth was increased from 5 MHz to 40 MHz.

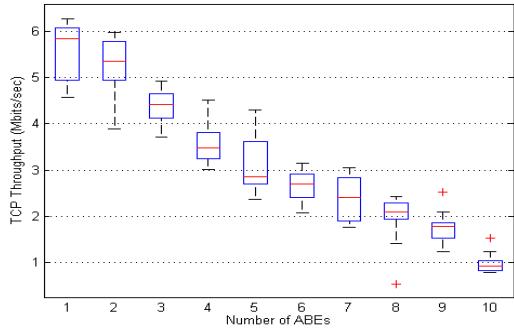


Figure 7: TCP Throughput versus Number of ABEs

Packet loss and jitter were also measured in the secondary P2MP network using iPerf and UDP traffic. The results are

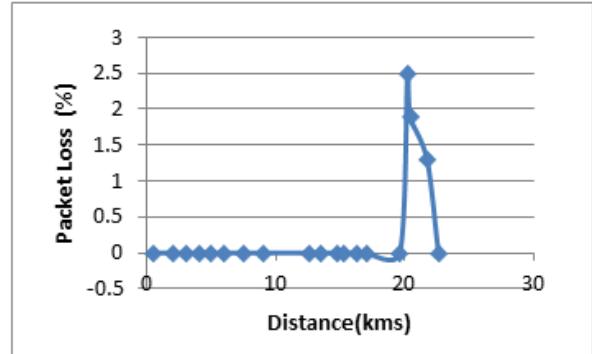


Figure 8: UDP Packet Loss versus Distance

shown Figure 8 and Figure 9. No packet loss was observed up to a distance of 20 km. Thereafter, it went up to 2.5% at a distance of 22 km. The rise was quite steep but was still a fairly small percentage at 2.5%. The jitter values were below 10 ms up to a distance of 14 km and then went up fairly steeply to 30-50 ms at 20 km and beyond.

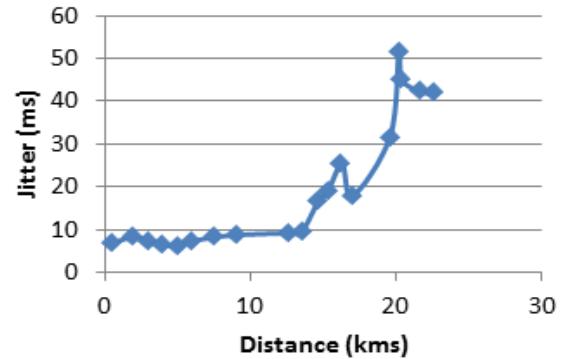


Figure 9: UDP Jitter versus Distance

Note that all the field trials were carried out by setting the channel bandwidth at 5 MHz. By increasing this to 20 MHz, the throughput can be increased at the cost of slightly reduced network range. The channel bandwidth can be varied dynamically based on the demands of the network and the number of stations connected to the base station. However, this needs to be done in a controlled fashion in order to ensure that all stations are in sync.

The ABE used in the field trial has a directional antenna, which is a built-in sector antenna with a sector angle of 40-50 degrees. This means that the ABE needs to be reoriented periodically to align with the base station. In addition, it needs to be stabilized against the rocking movements of the boat due to the waves in the ocean. If it is not stabilized, it could lead to transient link loss.

During the field trials, the reorientation was done manually and the sea state was fairly calm and hence the rocking of the boat was not a big problem. Work is underway to build a mechanical automatic reorientation and stabilization system. In the long term, this will be done electrically with beam forming using antenna arrays. The goal is to develop this indigenously in order to make it cost-effective.

Other marine networking solutions field tested in the past [17] [8] [7] have used older technology and also a mesh network architecture. For these reasons, the results obtained by them have not been as impressive. They achieved a maximum range of 11-15 km only.

## 4. CONCLUSIONS

Tremendous progress has been made towards realizing a cost-effective solution for the vexing problem of isolation of marine fishermen for several days together from the mainland during their fishing trips. Our field trials over the Arabian Sea have demonstrated that by using a multi-level hierarchy of extremely mobile infrastructure networks, it is possible to achieve Internet coverage over a wide range from the shores, as far away as 120 km. While the primary P2MP network can provide a range of 40-45 km, the secondary P2MP networks using boats opportunistically as mobile base stations can extend the range by an additional 15-20 km. Since the fishing zones tend to be uniformly distributed, it is reasonable to assume that there will be boats available 15-20 km apart to extend the network range to the desired 120+ km.

This is a classic example of scientific research resulting in a tangible benefit to an underprivileged section of the society. A pilot deployment of the proposed solution is currently in progress among the fishing community in a coastal village in Kerala, India. This will help us to further validate the solution and gain some valuable operational experience. It will also help in testing the durability and ease of use of the proposed solution. Once the pilot deployment is successfully completed, we will partner with an appropriate entity - an enterprise, an NGO or a governing body, to roll out this solution on a larger scale.

## 5. ACKNOWLEDGMENTS

Our chancellor, Dr. Mata Amritanandamayi Devi, a world renowned humanitarian leader popularly known as Amma, was instrumental in raising our awareness to this real problem among the fishing community. She also provided the inspiration and guidance for finding a viable solution to the problem. We wish to express our gratitude to her. This project is partly funded by Information Technology Research Agency (ITRA) under the Ministry of Electronics and Information Technology (MeitY), Government of India (GoI).

## 6. REFERENCES

- [1] K P Biswas. Industrial fisheries. 1, January 2004.
- [2] Handbook on fisheries statistics.
- [3] K. A. Unnikrishna Menon, V. N. Menon, and R. D. Aryadevi. A novel approach for avoiding water vessel collisions using passive acoustic localization. In *2013 International Conference on Communication and Signal Processing*, pages 802–806, April 2013.
- [4] H. S Jennath, M. K Anju, D. Raj, and S. Rao. Comparative study of backhaul options for communication at sea. In Dr. Rajesh K Bhatia and Dr. Janahanlal Stephen, editors, *Sixth International Conference on Recent Trends in Information, Telecommunication and Computing*, pages 69–83, Chennai, India, 2015. McGraw-Hill.
- [5] Kameswari Chebrolu, Bhaskaran Raman, and Sayandeep Sen. Long-distance 802.11b links: Performance measurements and experience. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*, MobiCom '06, pages 74–85, New York, NY, USA, 2006. ACM.
- [6] Rabin Patra, Sergiu Nedevschi, Sonesh Surana, Anmol Sheth, Lakshminarayanan Subramanian, and Eric Brewer. Wildnet: Design and implementation of high performance wifi based long distance networks. In *Proceedings of the 4th USENIX Conference on Networked Systems Design & Implementation*, NSDI'07, pages 7–7, Berkeley, CA, USA, 2007. USENIX Association.
- [7] M. t. Zhou, V. D. Hoang, H. Harada, J. S. Pathmasuntharam, H. Wang, P. y. Kong, C. w. Ang, Y. Ge, and S. Wen. Triton: high-speed maritime wireless mesh network. *IEEE Wireless Communications*, 20(5):134–142, October 2013.
- [8] Marcom: Broadband at sea, internet for coast, polar regions, offshore and sea farming. Online.
- [9] Bluecomplus. Connecting humans and systems at remote ocean areas using cost-effective broad-band communications. *Factsheet*, page 1, 2016.
- [10] SpeedCast. Maritime - two-way satellite service, online. *Factsheet*, page 1, 2016.
- [11] AxxessMArine. Axxess marine revolutionary vsat solutions. *Factsheet*, page 1, 2016.
- [12] S. Unni, D. Raj, K. Sasidhar, and S. Rao. Performance measurement and analysis of long range wi-fi network for over-the-sea communication. In *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), 2015 13th International Symposium on*, pages 36–41, May 2015.
- [13] S. N. Rao, D. Raj, S. Aiswarya, and S. Unni. Realizing cost-effective marine internet for fishermen. In *2016 14th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pages 1–5, May 2016.
- [14] B. Aswini, D. Raj, K. Sasidhar, and S. Rao. A network operations center for a communication network of fishing vessels at sea. *Sixth International Conference on Advances in Communication, Network, and Computing* àÁš CNC, Chennai, India, 5(3):39, 2015.
- [15] Ubiquiti Networks. *rocketm*, June 2016.
- [16] Ubiquiti Networks. *nanostationm*, June 2016.
- [17] N. Fuke, K. Sugiyama, and H. Shinonaga. Long-range oversea wireless network using 2.4 ghz wireless lan installation and performance. In *Computer Communications and Networks, 2003. ICCCN 2003. Proceedings. The 12th International Conference on*, pages 351–356, Oct 2003.

# On Diameter Based Community Structure Identification in Networks

Arnab Kumar Ghoshal  
 Department of Computer Science  
 Shibpur Dinobundhoo Institution (College)  
 Howrah, India  
 mr.arnabghoshal@gmail.com

Nabanita Das  
 Advanced Computing and Microelectronics Unit  
 Indian Statistical Institute  
 Kolkata, India  
 ndas@isical.ac.in

## ABSTRACT

Several community detection algorithms for large scale networks have been reported in the literature so far. But to the best of our knowledge none of these algorithms consider the diameter of the community as a key parameter. In this paper, we propose a new metric to measure the quality of the community structure identified in a network which is less computation intensive and more realistic. Also, we develop a new heuristic to identify community structures considering the diameter of the community as one of the determinant factors. To ensure good quality of the community structure, we restrict the diameter of a community, upper bounded by a certain value  $k \ll D$ , the diameter of the original network. Comparison with existing algorithms by simulation on well-known graphs such as Zachary karate club, Dolphin network and Football club shows our algorithm performs better in terms of modularity of the community structure achieved. Also, it establishes the fact that the modularity parameter defined by us, can compare the community structures of a given network better.

## CCS Concepts

- Networks → *Online social networks;*

## Keywords

Large Networks, Internet of Things (IoT), Community Structure, Diameter, Modularity, Good Quality Communities

## 1. INTRODUCTION

Recently, with the emergence of many large scale complex networks like social networks, biological networks, internet application networks, network of IoT, and many more, the analysis and characterization of these networks are becoming important. The term *community structure* in graphs generally means natural division of the nodes into densely connected subgroups such that connections between subgroups are sparse, compared to the connections within the same subgroup. The identification and analysis of such communities can provide immense knowledge in understanding and characterizing the related network and its properties.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ICDCN '17, January 04-07, 2017, Hyderabad, India

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3018285>

In the modern era of computing, in order to make every physical object intelligent and capable for human interaction a new computing concept, called Internet of Things (IoT) is introduced, where we can connect all physical objects through wireless signals. The connection of these physical objects forms a large complex network. Identifying community structures in such large IoT network, in fact, can improve communication mechanism, power consumption and resource sharing of every IoT device to achieve high quality data integration to offer better QoS.

It is evident that communities in a graph are loosely defined and also it is hard to compare the qualities of two community structures generated by different algorithms out of the same graph [1]. So far the quality of the community structure is measured by modularity metric as proposed by Newman in [1] which is based on the expected connectivity between nodes, as has been explained in the next section. A high modularity value indicates good quality community structure. Many algorithms have been proposed so far based on Newman's modularity metric [1] to detect community structures in a network [2–12]. Given a graph  $G(V, E)$ , the problem of community detection is to compute a partitioning of the nodes that maximizes modularity which is NP-complete [13]. However, the more fundamental issue is that how to define the modularity metric properly so that it may compare the quality of two solutions for the same graph in an unambiguous way.

In this paper, we consider a new approach to compute modularity of a community structure based on the topology of the given graph itself, not on the expected connectivity, as has been done in [3]. We first find the node modularity defined as the ratio of the number of neighbors of a node  $v$  within the same community to the total degree of the node  $v$ . Modularity of a community structure is then calculated as the average node modularity within it. With this definition, the value of the node modularity varies between 0 and 1. By our approach, a node always attempts to maximize its modularity by joining a community. Essentially, this may lead to form a single community which is the graph itself. To limit the addition of a node into a community, a restriction is set on the diameter of the community, such that it is bounded by a predetermined constant  $k$ ,  $k \ll D$ , where  $D$  is the diameter of the original graph  $G(V, E)$ . Based on this criteria, we propose an algorithm for partitioning a given graph into some communities such that the modularity of the community structure is maximized. We apply our algorithm on well-known graphs to compare the quality of the communities achieved by our algorithm with other existing algorithms. It shows improvement in modularity according to our definition compared to the definition presented in [3] as well. Also, it reveals that our proposed modularity definition can compare the community structures better.

The rest of the paper is organized as follows. Section 2 presents the literature review. Preliminaries are presented in Section 3. Sec-

tion 4 presents the proposed algorithm. The simulation and test results are included in Section 5 and finally, Section 6 concludes the paper with the scope of future work.

## 2. LITERATURE REVIEW

So far, the community detection problem for large networks have been extensively studied in the literature.

Newman et al. [1] first proposed the algorithm for community detection based on edge betweenness centrality computation where they have found the community structures by repeatedly removing the edges with highest betweenness centrality. But the computation time of this divisive approach is  $O(m^2n)$ , where  $m$  and  $n$  are the number of edges and nodes respectively. Also they have proposed a metric, called modularity, in order to measure the quality of a community structure by subtracting the expected number of intra-community edges from the actual number of intra-community edges.

In order to improve the computation time Newman in [2] have proposed an agglomerative hierarchical clustering method. In agglomerative hierarchical clustering, each node of a graph is first treated as a single community and then the nodes are being grouped to form community structures where they have maximum number of neighbors. Thus the computation time for this approach becomes  $O(n.(n + m))$ . Similarly, Clauset et al. [3] have proposed a community detection algorithm based on agglomerative hierarchy in  $O(m.d.\log n)$  time, where  $m$  and  $n$  are the number of edges and nodes respectively and  $d$  is the depth of the community structure. Raghavan et al. [4] have proposed a label propagation algorithm in order to detect community structures in near linear time. In their approach, they have initialized each node with a unique label and at each step, allowed them to adopt the label that most of their neighbors currently have. At the end of the algorithm, nodes having same labels are grouped together as communities [4].

In [5], Blondel et al. have proposed a modularity optimization based heuristic method to detect community structures in linear time that is extremely fast. Initially, each node is treated as a single community. So, the number of communities is equal to the number of nodes. Next for each node they have considered the neighbors and evaluated the gain in modularity by removing the node from its community and placing it in the community of its neighbors. The node is placed in the community where maximum gain in modularity is achieved. This process has continued until no further improvement is possible. After that the communities found are treated as a single node for the next phase and the same process is repeated. The process stops when there is no improvement for the next phase.

For faster computation, parallel algorithms for community detection have been studied extensively. Riedy et al. [6] have presented a parallel agglomerative algorithm for community detection, and implemented it on both the Cray XMT2 and OpenMP platforms. The algorithm begins by taking each node as a community and then neighboring communities are merged to maximize the modularity. The number of operations required by this algorithm is  $O(nm)$ . They achieved a speedup upto  $13\times$  on a four processor, 40-physical-core Intel-based platform. In [7], Soman et al. have designed a parallel version of [4] in order to detect community structure for large scale networks on both multi-core architecture and GPU platform with a computation time of  $O(m.(k + d))$ , where  $m$  is the number of edges,  $d$  is the average degree of a node and  $k$  is the number of iteration which is around 10 for massive graphs. Newman in [8] has proposed a spectral algorithm that uses the *Eigen vectors* of the normalized Laplacian matrix representation in order to detect community structures. Cheong et al. [9] have implemented the parallel version of [5] on multi-GPU platforms.

They have achieved  $3\times$  speedup on an average over Louvain [5] on single GPU platform, and another additional  $2\times$  speedup on average on multi-GPU platform. In [10], Lu et al. have proposed a heuristic to parallelize Louvain [5] method on multithreaded architecture. Cheng et al. [11] have proposed a divisive spectral method for community detection. The technique first uses a sparsification operation and then uses a repeated bisection spectral algorithm to find the community structures. Held et al. [12] have proposed a community detection algorithm for dynamic graphs using nearest hubs. Though the modularity achieved by this algorithm is not as high as Louvain [5] but it is better than spectral clustering [8].

With lots of research reported so far on designing efficient algorithms for community detection algorithms for large graphs, it is interesting to note that a unique quantitative measure of the quality of the achieved community structure is yet to come up that is capable to compare the quality of proposed solutions unambiguously.

## 3. PRELIMINARIES

As has been mentioned above, community detection in large scale networks based on edge betweenness centrality was first proposed by Girvan and Newman [1]. After that several algorithms have been proposed in order to reduce the computational complexity and to improve the modularity measure [2–12]. All these algorithms measure modularity by subtracting the expected number of intra-community edges from the actual number of intra-community edges [3] which can be written mathematically as:

$$Q = \frac{1}{2m} \sum_{uv} \left[ A_{uv} - \frac{d_u d_v}{2m} \right] \delta(c_u, c_v) \quad (1)$$

where  $A_{uv}$  is an element of the adjacency matrix of the graph,  $c_u$  denotes the community of  $u$ ,  $\delta$ -function  $\delta(c_u, c_v)$  is 1 if  $c_u = c_v$ , i.e., both  $u$  and  $v$  belong to the same community, and 0 otherwise,  $m = \frac{1}{2} \sum_{uv} A_{uv}$  is the total number of edges in the graph and the degree  $d_u$  of a node  $u$  is defined as number of edges incident upon it:

$$d_u = \sum_v A_{uv} \quad (2)$$

In this paper, we define the modularity of a community structure based on the properties of the given graph itself, not on the expected connectivity as in [3]. But it is evident that given a graph  $G(V, E)$ , the modularity is to be defined based on the existing edges which is the real scenario, instead of defining it with reference to the expected connectivity. In order to provide a more realistic definition for community modularity, we first define modularity of a node  $v$ . Let  $C$  be a community and  $d_v^{in}$  denote the number of edges of a node  $v \in C$  incident on other nodes in  $C$ . The node modularity is defined as

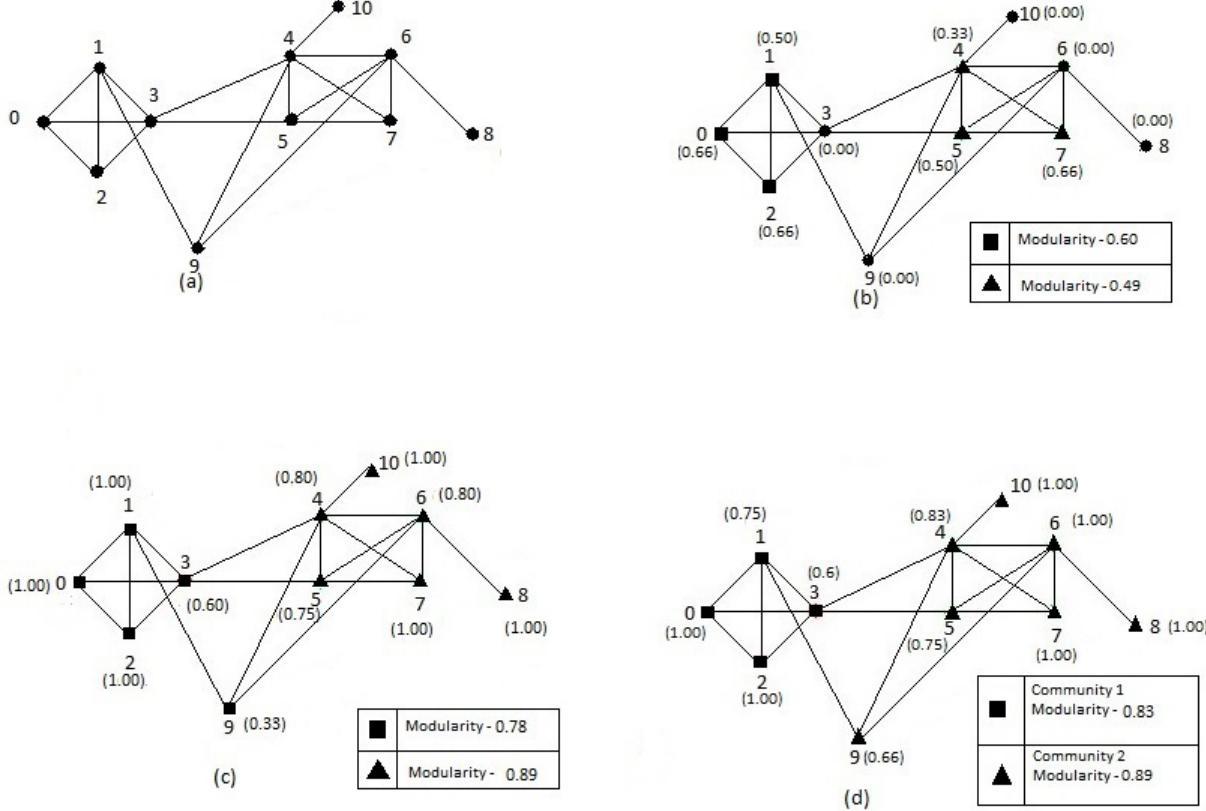
$$v_{mod} = \frac{d_v^{in}}{d_v} \quad (3)$$

Now we define the community modularity as

$$c_{mod} = \frac{\sum_{\forall v \in C} v_{mod}}{n_c} \quad (4)$$

where  $n_c$  denotes the total number of nodes in community  $C$ . This modularity definition has two advantages - i) it is computationally less expensive compared to [3] and also computable in distributed fashion by individual nodes, and ii) we only consider the actual number of edges existing within a community  $C$  instead of using the expected one to address the real scenario.

With this new definition, the modularity value varies within the



**Figure 1:** (a) Example Graph (b) All disjoint 3-cliques for the Example Graph (c) Intermediate steps (d) Communities of the Example Graph

range from 0 to 1 as we mentioned earlier. We propose an iterative procedure to generate a good quality community structure, every-time by including either an isolated node into a community, or migrating a node from one community to other, to improve the average modularity of the community structure unless the diameter of the community exceeds a predefined value  $k$ . Normally  $k \ll D$ , where  $D$  is the diameter of the original graph. Our criterion is that good quality community structure will be such that not only the modularity will be close to unity, but at the same time the diameter of each community should be small to maintain good connectivity among the intra community nodes.  $M$  is the overall community modularity which is calculated by taking the average of the modularities of the community structures.

#### 4. PROPOSED ALGORITHM

In our proposed algorithm, we start from an initial solution, and at each iteration allow a single node to migrate from one community to another that results maximum improvement in average modularity of the solution keeping the diameter of each community bounded by  $k$ . For a node  $v$ , let  $\Gamma(v)$  denote the list of neighbors of  $v$ , sorted in ascending order of degree. In **Algorithm 1**, we describe the procedure of finding the initial set of components which are nothing but a set of disjoint 3-cliques, the simplest structures with diameter 1. In case, the initial component set is empty i.e.  $C = \emptyset$ , every individual node of the graph is considered to be a community. Next, we start finding the community structures iteratively using the procedure described in **Algorithm 2**.

For the graph  $G(V, E)$  shown in Fig. 1(a), we first compute the

---

#### Algorithm 1 Initial set of components algorithm

---

**INPUT:** Graph:  $G(V, E)$ ,  $\Gamma(v)$

**OUTPUT:** A set of components :  $C = \{c_1, c_2, \dots, c_l\}$

sort the nodes  $V = \{v_1, v_2, \dots, v_n\}$  in increasing order of their degree

**for each**  $i \in V$  **do**

**if**  $i$  is not marked **then**

**for each**  $j \in \Gamma(i)$  **do**

**if**  $j$  is not marked **then**

**if**  $i > j$  **then**

$k \leftarrow$  find a common element of  $\Gamma(i)$  and  $\Gamma(j)$

**if**  $k \neq$  **then**

**if**  $k$  is not marked **then**

                        make 3-clique  $< i, j, k >$  and mark  $i, j, k$ .

                        set the diameter of the 3-clique to 1

---

initial set of components i.e. the disjoint set of 3-cliques shown in Fig. 1(b). Here we can see that the example graph has two components at the initial level. Now for each initial component, we first compute the node modularity  $v_{mod}$  using equation (3) for every node within it and then the community modularity  $c_{mod}$  using equation (4). The node modularity and community modularity values obtained are shown in Fig 1(b). Next, for the proper placement of the nodes, we follow the steps of **Algorithm 2** where we process each node of the graph in turn and move it to the component where

---

**Algorithm 2** Community detection algorithm

---

**INPUT:** Graph:  $G(V, E)$ ; Initial Component set:  $C = \{c_1, c_2, \dots, c_l\}$ ; Threshold Diameter:  $k$   
**OUTPUT:** Final set of components called communities

```
iter ← 1
while iter ≤ MAX_ITERATION do
    for each  $v \in V$  do
        find the component  $c_x \in C$  for maximum increment in
        community modularity with diameter is less than equal
        to  $k$ . In case of tie, choose one with minimum diameter

        if  $flag(v) = 1$  and  $v \in c_y$  then
            if the transition of  $v$  from  $c_y$  to  $c_x$  improves  $M$  then
                if diameter of  $c_x \leq k$  then
                     $v$  joins  $c_x$ 
                if transition of  $v$  disconnects some nodes in  $c_y$  then
                    remove those nodes from  $c_y$  and set the flag
                    values to 0
                    update the diameter and community modularity of  $c_y$ 
                    and  $c_x$ 
                else
                     $v$  joins  $c_x$ 
                     $flag(v) \leftarrow 1$ 
                    update the diameter and community modularity of  $c_x$ 
            if no node joins any of the component then
                return
```

---

maximum increment in modularity is achieved and also the diameter of the component remains less than equal to  $k$ . In Fig 1(b), we have two initial components  $< 0, 1, 2 >$  and  $< 4, 5, 7 >$ . Here node 0, 1 and 2 are in the component 1. Similarly, node 4, 5 and 7 are in component 2. In Fig 1(c), joining of node 3 and 9 in component 1 results in maximum increment in modularity. Similarly, joining of node 6, 8 and 10 in component 2 results in maximum enhancement in modularity. Now we can see that the transition of node 9 from component 1 to component 2 improves overall community modularity  $M$  (that is obtained by taking the average of the modularities of the community structures). So, we allow this transition. We have shown this in Fig. 1(d) which is the final result where each component is termed as a community. We assume  $k = 2$  for the example graph.

## 5. SIMULATION AND TESTING

We have implemented our algorithm in  $C$  language on a 64 bit machine with 4 GB RAM and GNU/Linux Operating System (OS). We have used heap sort algorithm for sorting the nodes in **Algorithm 1** and implemented the algorithm in [14] for diameter computation. To compute final community modularity, we have taken the average of the modularity values for all the resulting communities. Comparison is done with the serial Louvain algorithm<sup>1</sup> as it is the most successful community detection algorithm in the state of the art till now and a python implementation of GN [1] algorithm<sup>2</sup> which is the pioneer in this area. We have used the real world small data sets of Zachary Karate Club [15] (34 nodes, 78 edges), US College Football Network [16] (115 nodes, 1230 edges) and Dolphin's Network [17] (62 nodes, 159 edges) to test and simulate our algorithm. The reason behind choosing these data sets is that they are the most widely adopted [7] benchmarks [8] with limited size,

<sup>1</sup><https://sites.google.com/site/findcommunities/>

<sup>2</sup><https://github.com/kjahan/community>

and has been used widely in recent day research [11, 12, 18]. These data sets are taken from the UCI Network Data Repository<sup>3</sup>. As the diameter of all small world graphs is found to be less than equal to 6 and also as we start exploring communities from the initial set of components with diameter 1, we have studied the solutions varying the threshold value for the community diameter  $k$  within the range from 2 to 5. We have also set the value of  $MAX\_ITERATION$  to a large value. Here,  $N$  denotes the number of communities in the final solution.

**Table 1: Number of communities and the modularity for  $2 \leq k \leq 5$  according to our definition**

k	Karate		Dolphin		Football	
	N	Modularity	N	Modularity	N	Modularity
2	2	0.78	9	0.34	12	0.61
3	2	0.91	4	0.72	10	0.63
4	2	0.91	4	0.78	9	0.69
5	2	0.91	4	0.79	9	0.69

Table 1 shows the number of communities  $N$  and the modularity values  $M$  (according to our definition) for the considered data sets for different values of  $k$ . It shows that with the increase in  $k$ , initially  $M$  increases rapidly, as is expected, but it gets almost saturated for  $k \geq 4$ . On the contrary,  $N$  decreases with increasing  $k$ , and becomes almost invariant for  $k \geq 4$  which is evident. It is obvious that increase in  $k$  also will demand more computation, and more time to converge.

So, we have compared the community modularity values of our algorithm for only  $k = 4$  with GN [1] and Louvain [5]. In Table 2, we have shown the comparison results with  $k = 4$ , according to our modularity definition where we can see that the community modularity given by our algorithm is always better or comparable even with smaller number of communities.

**Table 2: Comparison of number of communities  $N$  and the modularity  $M$  according to our definition**

Algorithm	Karate		Dolphin		Football	
	N	Modularity	N	Modularity	N	Modularity
GN	2	0.91	2	0.50	12	0.58
Louvain	4	0.76	10	0.58	10	0.70
Our Algo.	2	0.91	4	0.78	9	0.69

Again if we look at  $N$ , we see that for Karate club, our algorithm gives the ground truth reality with  $N = 2$ , which is same as GN. But at the same time, Louvain gives 4 communities for the Karate club and also the community modularity is less than that obtained by GN. For Dolphin's network, by our algorithm  $N = 4$  which is more than that given by GN but less than that achieved by Louvain. For Football club, number of communities given by Louvain method is less than that achieved by the GN method and also the community modularity is same for those two algorithms. In case of our algorithm, number of communities is less than that by GN and Louvain method with community modularity closer to the Louvain method and at the same time it is greater than that achieved by the GN method. This contradicts the intuition that more is the number of partitions better will be the community modularity.

Table 3 shows the comparison according to Newman's modularity definition where we can see that the modularity value given by

<sup>3</sup><https://networkdata.ics.uci.edu/index.php>

**Table 3: Comparison of number of communities and the modularity according to Newman’s modularity definition**

Algorithm	Karate		Dolphin		Football	
	N	Modularity	N	Modularity	N	Modularity
GN	2	0.41	2	0.49	12	0.60
Louvain	4	0.41	10	0.51	10	0.60
Our Algo.	2	0.41	4	0.55	9	0.63

our algorithm is either same or better than others for the considered data sets. We see that for Karate club all the algorithms including ours produce the same value of  $M$ , though the values of  $N$  are different. For Dolphin’s network, the number of communities given by our algorithm is 4 which is more than number of communities given by GN but less than the number of communities given by Louvain. But the important thing to notice is that community modularity given by our algorithm is much better than obtained by GN and Louvain method. For Football club, number of communities given by Louvain method is less than that obtained by GN method and also the community modularity is same for those two algorithms. In case of our algorithm, number of communities is less than that obtained by GN and Louvain method but interestingly community modularity is higher. This is also against the general expectation that with more number of partitions, the community modularity will be higher [7]. So, it is evident that for the given graphs, our algorithm is more successful to find close-knit smaller partitions with bounded diameter.

Moreover, if we look at the results on Table 3, we observe that for both Karate and Football club though the number of communities are different achieved by GN and Louvain algorithms, but the modularity values are same. So, it is not possible to compare the quality of two solutions for such graphs considering the modularity values only as defined by Newman. At the same time if we look at the results on Table 2 which is based on our unique modularity definition, we can easily compare between the quality of two solutions based on the modularity value which says that for Karate club, GN performs better, whereas, for Football club, Louvain results better community structures with higher modularity. In that sense, our definition is more successful to compare the quality of two solutions.

## 6. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a new modularity definition for the community structure, and designed a novel sequential diameter based community detection algorithm in order to find the community structures within a network. We see that when the diameter of the community is  $k = 4$ , we get better modularity values on an average. Our modularity definition is based on the real graph and can be easily computed in a distributed fashion. So far, we have simulated our algorithm on small real world network data sets and compared it with existing algorithms GN [1] and Louvain [5]. Simulation results show that our algorithm produces same or better quality community structures, and also our proposed definition of modularity can compare the quality of community structures better.

Extensive simulation studies are yet to be done on large real world network data sets and also on large scale random graphs in order to evaluate the performance of our proposed algorithm, and the new definition of modularity. It is true that for large networks, the implementation will demand extensive computation. Hence going forward we would like to parallelize our algorithm in order to reduce its computation time.

## 7. REFERENCES

- [1] M. E. Newman and M. Girvan, “Finding and evaluating community structure in networks,” *Physical review E*, vol. 69, no. 2, p. 026113, 2004.
- [2] M. E. Newman, “Fast algorithm for detecting community structure in networks,” *Physical review E*, vol. 69, no. 6, p. 066133, 2004.
- [3] A. Clauset, M. E. Newman, and C. Moore, “Finding community structure in very large networks,” *Physical review E*, vol. 70, no. 6, p. 066111, 2004.
- [4] U. N. Raghavan, R. Albert, and S. Kumara, “Near linear time algorithm to detect community structures in large-scale networks,” *Physical Review E*, vol. 76, no. 3, p. 036106, 2007.
- [5] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, “Fast unfolding of communities in large networks,” *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.
- [6] E. J. Riedy, H. Meyerhenke, D. Ediger, and D. A. Bader, “Parallel community detection for massive graphs,” in *International Conference on Parallel Processing and Applied Mathematics*. Springer, 2011, pp. 286–296.
- [7] J. Soman and A. Narang, “Fast community detection algorithm with gpus and multicore architectures,” in *Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International*. IEEE, 2011, pp. 568–579.
- [8] M. E. Newman, “Spectral methods for community detection and graph partitioning,” *Physical Review E*, vol. 88, no. 4, p. 042822, 2013.
- [9] C. Y. Cheong, H. P. Huynh, D. Lo, and R. S. M. Goh, “Hierarchical parallel algorithm for modularity-based community detection using gpus,” in *European Conference on Parallel Processing*. Springer, 2013, pp. 775–787.
- [10] H. Lu, M. Halappanavar, and A. Kalyanaraman, “Parallel heuristics for scalable community detection,” *Parallel Computing*, vol. 47, pp. 19–37, 2015.
- [11] J. Cheng, L. Li, M. Leng, W. Lu, Y. Yao, and X. Chen, “A divisive spectral method for network community detection,” *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2016, no. 3, p. 033403, 2016.
- [12] P. Held and R. Kruse, “Online community detection by using nearest hubs,” *arXiv preprint arXiv:1601.06527*, 2016.
- [13] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, and D. Wagner, “On modularity clustering,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 20, no. 2, pp. 172–188, 2008.
- [14] Y. Fujiwara, M. Onizuka, and M. Kitsuregawa, “Real-time diameter monitoring for time-evolving graphs,” in *Database Systems for Advanced Applications*. Springer, 2011, pp. 311–325.
- [15] W. W. Zachary, “An information flow model for conflict and fission in small groups,” *Journal of anthropological research*, pp. 452–473, 1977.
- [16] M. Girvan and M. E. Newman, “Community structure in social and biological networks,” *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.
- [17] D. Lusseau, “The emergent properties of a dolphin social network,” *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 270, no. Suppl 2, pp. S186–S188, 2003.
- [18] G. Li, D. Zhang, K. Xie, T. Huang, and Y. Li, “A gpu based fast community detection implementation for social

network,” in *International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 2015, pp. 688–701.

# Towards Smart City: Sensing Air Quality in City based on Opportunistic Crowd-sensing

Joy Dutta, Chandreyee Chowdhury, Sarbani Roy, Asif Iqbal Middya, Firoj Gazi

Department of Computer Science and Engineering, Jadavpur University, Kolkata-700032, India

joy.dutta.in@ieee.org, {chandreyee, sarbani.roy}@cse.jdvu.ac.in, asif.md.ju@gmail.com, firojgazi123@gmail.com

## ABSTRACT

Cities are expanding and more and more citizens are exposed to air pollutants both indoors and outdoors. This may have adverse effects on citizens' health. In this paper, we present AirSense, an opportunistic crowd-sensing based air quality monitoring system, aimed at collecting and aggregating sensor data to monitor air pollution in the vicinity (building/neighbourhood) and the city. We introduce a light weight, low power and low cost air quality monitoring device (AQMD) and demonstrate how AQMD and smartphones in a crowd collaboratively gather and share data of interest to the cloud. In cloud, collected data are analyzed and an aggregate view is generated from data collected from various sensors and from different users for providing an air pollution heat map of the city. Unlike previous works, both micro and macro level air quality monitoring is possible with Airsense. End user can view his/her pollution footprint for the whole day, the neighborhood (local) air quality and AQImap (air quality index map) of the city on his/her smartphone. The system is implemented and the prototype is also evaluated.

## Categories and Subject Descriptors

C.3 [Special-Purpose And Application-Based Systems]

## General Terms

Crowd-sensing, sensors, cloud, air pollution.

## Keywords

Air quality; Opportunistic crowd-sensing; Sensing device

## 1. INTRODUCTION

Presently, air pollution is a global concern as it may cause many chronic and fatal diseases. Especially in developing countries, rapid urbanization is happening, without paying attention to the environment. As urban areas have high density of population, maintaining air quality is becoming more and more challenging. People are unknowingly exposed to harmful gases making them prone to many deadly diseases. Not only prolonged exposure to polluted air, some gasses if inhaled even for a short time can cause serious illness. The effect is even more severe in young and elder adults.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [Permissions@acm.org](mailto:Permissions@acm.org).

*ICDCN '17*, January 04-07, 2017, Hyderabad, India

© 2017 ACM ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3018286>

However, in reality, there are inadequate air quality measurement stations [2] in a city of a developing country. The reason behind this is the appreciable cost of building and maintaining such a station. On the other hand, air quality readings taken from these stations are highly reliable and accurate as measured using carefully calibrated professional equipment. U-Air system proposed in [2] analyzes readings reported by a few air quality monitoring stations and a variety of data sources including meteorology, traffic flow etc. observed in the city. It uses artificial neural network to find spatial correlation of air pollution in different parts of the city. Some effort is being made for designing portable sensors, that can be attached to public vehicles[4] so that data can be collected on the move. This little bit saves the maintenance effort. But, these air quality monitoring systems are able to view the effects on a macro scale. Moreover, indoor air quality monitoring, (for example in factory or laboratory) are important as a person stays most of his/her time indoor. Nowadays, a few devices are being made that can be placed at specific locations indoor to detect air quality. These again provides a macro scale view of indoor air pollution. For home, it may suffice but for factories and/or laboratories, these may prove to be inadequate. More importantly, individual's pollution footprint cannot be measured and its associated health hazards cannot be assessed with these macro scale devices. Few research efforts are being made in this direction that uses low cost portable sensing devices that interfaces with smartphones to be carried by citizens. The devices sense data with or without user intervention and send it to cloud through the Internet. The advantage of these group of crowd-sensing based air pollution monitoring applications is ease of maintenance and low cost of procuring these devices. Some relevant air quality monitoring solutions based on participatory sensing approach have been described in [3-6]. In participatory sensing the user decides when to sense and send data depending on a number of factors including incentive, remaining energy of the device etc. In [7], wearable sensors are used to sense air quality that is pushed to the cloud through the smartphone. In [9-10] efforts are being made where devices are carried by citizens for sensing and the data collected by citizens can be shared through social media to spread awareness. However, these works mainly focus on outdoor air quality monitoring mainly of public places in the city. However, most citizens stay indoor or in campuses most of their time in a day. Consequently, the main objective of this work is to design and develop a system called *AirSense* for air quality monitoring of our neighborhood both indoor and outdoor. The proposed system will encourage the citizens to participate in a crowd-sensing initiative, which could be a backbone of any smart city. The nature of the crowd-sensing used in this work is categorized as opportunistic sensing [1] where, user involvement is minimal, which generally ensures reliable data at regular intervals. Nowadays, smartphones and tablets are increasingly becoming an essential part of human life as the most effective and convenient communication tools not bounded by time and space. Hence, a portable Air Quality

Monitoring Device (AQMD) is designed that interfaces with smart handhelds to communicate data to the cloud.

AirSense can be used in two scenarios, *personal* and *collective* sensing, based on the circumstances being experienced. With the help of collective sensing paradigm, AirSense can provide air quality data with enhanced temporal and spatial resolution. Thus, AirSense can monitor air pollution of the entire city at low cost. Both indoor and outdoor monitoring can be performed seamlessly. This increases opportunities for the study of fine grained air quality monitoring and its impact on citizen's health. As opposed to traditional, expensive, heavy but highly accurate air quality measurements [2], the use of AirSense based on modern, low-cost and light weight sensors is very appropriate in today's world. The most important issue in obtaining accurate and spatially resolved air quality data is achieved through the participation of people in AirSense.

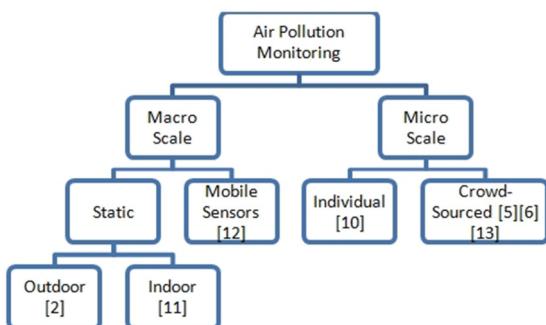
Though data is collected through registered user (volunteer) who carries the AQMD and a smart phone, registered users without carrying a device can see the Air Quality Index (AQI) of a nearby location and also the AQI map of its vicinity or the city through the AirSense application on their smartphone. Thus a person at a bus stop without carrying AQMD may get a fine grained pollution map if (s)he is a member of the crowd and hence can choose his/her route accordingly.

In contrast to existing works, both indoor and outdoor air quality can be measured through it. AirSense system also provides an opportunistic crowd-sensing approach for real-time notification of air quality in the neighborhood as well as the city to end users. Sensing and data collection processes are more flexible as opportunistic crowd-sensing is in place.

The paper is organized as follows. Related works is discussed in Section 2. Section 3 presents the AirSense system design in detail. The implementation outline is described in the next section followed by discussion on experimental results. Finally, Section 6 concludes.

## 2. RELATED WORK

Many research works are carried out on air pollution monitoring systems that are summarized in Fig. 1. In early 2000s, researches were carried out by deploying air quality sensors at major road crossings and important areas of a city. These sensors are very costly but produced accurate results. In [2] authors analyzed data from these sensors and investigated the effect of meteorological conditions, location, human mobility on air pollution to find a spatial correlation of air quality at various places of a city. They used artificial neural networks to classify various areas.



**Figure 1. Classification of state of the art techniques of air quality monitoring**

As the city limits are being extended, more and more population is shifted toward city, clean air is becoming an increasingly important concern not only outdoor but also indoor. Toward this, some devices [11] are being marketed recently, that incorporates lightweight sensing devices to be placed in rooms for quick monitoring of indoor air quality. However, many of these devices do not interface with Android enabled smart handhelds for performing more analysis and/or sharing the values. Another direction of research that is currently explored is mobile air quality sensors that are attached to public vehicles. As the vehicles move around the city, air quality data is collected [12]. This helps in easier maintenance than fixed monitoring stations. However, all these devices provide pollution monitoring at the macro scale. To assess an individual's pollution footprint, fine grained air quality monitoring is needed. Currently, some devices are made and provided as research prototypes like [10]. Few works are done on these custom designed devices that enables crowd-sensing/sourcing for large scale data collection with these personalized devices [4-6]. One of the first initiatives by using crowdsourcing for air quality monitoring applications was taken by the project MESSAGE (Mobile Environmental Sensing System Across Grid Environments) [14] from Cambridge University and partners in the UK. They developed low-cost fixed and portable devices and deployed in high densities in the Cambridge area. They found that, low cost devices when deployed in high densities, can give a much more accurate picture of the spatial and temporal structure of air quality in the urban environment. The devices can be mounted on vehicles. They focused on the city wide pollution footprint. In [13], developments by HazeWatch project is reported which is a crowd-sourcing based air quality monitoring application that aims to find an individual's pollution footprint. The lightweight monitoring devices designed by the project are mounted on private vehicles. The devices interface with smartphones to additionally send data to server so that data can be aggregated with other user's data to interpolate and have a city wide pollution map.

Most of the crowd-sensing applications designed till now are based on participatory sensing that enables the devices to send data when the user wants. For instance, in [4] a participatory sensing based system is developed where wearable sensors are used for air quality monitoring. The authors also designed a personalized real time notification mechanism for mobile application users. This gives users more control over participation in the crowd-sensing activity. However, task designers get more flexibility with opportunistic crowd-sensing as it provides reliable outcome at predictable intervals. This also enables more accurate data aggregation and analysis. Consequently, in this paper an opportunistic crowd-sensing based air quality monitoring system AirSense is designed and detailed.

## 3. AIRSENSE DESIGN

AirSense system is designed as a 4-tier architecture as shown in Fig. 2. Tier 1 constitutes of crowd who is providing data to AirSense and also responsible for consuming the services provided by AirSense. Tier 2 deals with data, and formatting of data for efficient transmission through the Internet. The communication between smartphones and cloud is handled in Tier 3 while Tier 4 deals with storage, aggregation and analysis of data. Finally, the result computed in Tier 4 is sent back to Tier 1 through Tier 3. Services can be pushed and/or pulled in AirSense.

Tier 1 describes the participation nature of people. We have considered two user categories.

Category	Description
C1	People with AQMD and Smartphone
C2	People with Smartphone only

C1 users can

- participate in crowdsensing activity,
- get their own pollution footprint and
- get AQIMap of the city

whereas C2 category users can only get city wise AQIMap that is the macro level view. Using AQI data provided by the local Air Quality Monitoring stations and C1 users, AQIMap is built for indoor as well as outdoor but the service can also be extended to users who do not have AQMD device so that they can also plan for a pollution free route.

As depicted in Fig. 2, **Tier 2** deals with formatting and cleaning air quality data before transmission. AQMD sends the data in the form of a tuple  $\langle \text{AQMD\_ID}, \{\text{sensed\_data}\} \rangle$  to nearby smartphone using the device's Bluetooth connection, whenever there is a significant change in the measurement of sensing data. Thus data transmission is not periodic but event driven in order to decrease network bandwidth under stable conditions. AQMD contains multiple gas sensors for monitoring air quality.

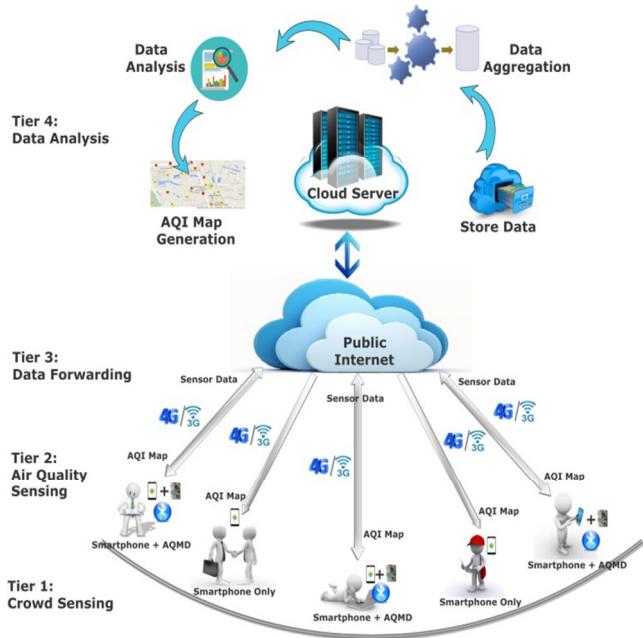


Figure 2: Architecture of AirSense

**Tier 3** is responsible for a data forwarding task. Data collected from the Smartphones or Tablet of AQMD users are transmitted to the cloud server through the Internet. Currently, all the modern smartphones are equipped with GPS navigation system. Thus GPS data is also combined with the tuple received from AQMD. This consolidated data is sent through the Internet to the cloud server. As AirSense fits in smart city applications, it is not unrealistic to assume that smartphones are connected to the Internet on the move both in indoor and outdoor scenarios either through WiFi or through 3G/4G connections. The sensed data forwarded through the Internet is stored, aggregated, analyzed at the cloud server in **Tier 4**. Here, in this application, two services are considered:

- generation of AQIMap on the basis of forwarded data from C1 users;
- building pollution footprint of individual users

Both C1 and C2 users may get the first service while the second one can be provided to C1 users only.

The database structure at the cloud end is shown in Fig. 3.

AirSense application on the smartphone is responsible to send entire information as shown in Fig. 3, in the tuple form  $\langle \text{Device\_ID}, \text{Location\_Type}, \{\text{Timestamp}\}, \text{Temperature}, \text{AQMD\_ID}, \{\text{sensed\_data}\}, \text{AQI}, \{\text{position}\} \rangle$ . Column 1 mentions Device\_ID which stands for MAC Address of the smartphone. This is a unique identifier. Column 2 represents Location\_Type which may have values like {indoor, atm, university, airport, bank, hospital, gym, museum, park,...etc.} as we found from Google Places API. Column 3 and 4 represent system's timestamp and temperature values which are collected using the OpenWeatherMap API in the corresponding fields of the database and is related to the device's location. Column 5 represents AQMD\_ID which is composed of Device\_ID and a random number which eventually gives us another unique no. corresponding to each AQMD. Column 6, 7, 8 and 9 represents the data set for sensed data which is composed of a popular and well accepted sensor set viz, MQ 135, MQ 2, MQ 7 and MQ9. They are sensitive for a range of gasses and can be used indoors at room temperature. Column 10 and 11 stands for the latitude and longitude of the device and are calculated from the device's GPS location. Then in Column 12, AQI is calculated by the norms of CPCB which is based on National Air Quality Indexing [15].

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	Device_ID	varchar(20)	latin1_swedish_ci		No	None	
2	Location_Type	varchar(10)	latin1_swedish_ci		No	None	
3	Time_Stamp	timestamp		on update CURRENT_TIMESTAMP	No	CURRENT_TIMESTAMP	ON UPDATE CURRENT_TIMESTAMP
4	Temp	float			No	None	
5	AQMD_ID	varchar(20)	latin1_swedish_ci		No	None	
6	MQ135_Data	int(10)			No	None	
7	MQ2_Data	int(10)			No	None	
8	MQ7_Data	int(10)			No	None	
9	MQ9_Data	int(10)			No	None	
10	Latitude	varchar(8)	latin1_swedish_ci		No	None	
11	Longitude	varchar(8)	latin1_swedish_ci		No	None	
12	AQI	int(10)			No	None	

Device_ID	Location_Type	Time_Stamp	Temp	AQMD_ID	MQ135_Data
MQ2_Data	MQ7_Data	MQ9_Data	Latitude	Longitude	AQI

Figure 3. Database Fields maintained in the cloud server by AirSense

Finally, data is stored in the cloud in **Tier 4** for further analysis as shown in Fig. 2. Data aggregation and analysis are the main responsibilities of this layer. Mobile-cloud computing emerges as a new computing paradigm where mobile devices exploit the cloud environment for both computation and storage purposes. Moreover, computation offloading to the cloud effectively expands the usability of mobile terminals beyond their physical

limits. Therefore, in AirSense, computation and storage tasks are offloaded to the cloud. AQImap of the city or given area can be generated from the aggregate data. Moreover, based on the user's data, individual's pollution footprint can also be formed.

## 4. IMPLEMENTATION OUTLINE

The AirSense platform consists of three key components, the AQMD, mobile application and the cloud service.

### 4.1 AQMD

The circuit diagram of AQMD is shown in Fig. 4 and a prototype of AQMD, which we built in our lab is deployed in real world scenario which is connected via Bluetooth with tab/ smartphone and is shown in Fig. 5. Though it is evident that more sensors (viz, PM<sub>10</sub>, PM<sub>2.5</sub>, O<sub>3</sub>, NO<sub>2</sub>, SO<sub>2</sub> etc.) we will be using, the more accurate data we will get but the cost of the device will be increased. So, to build the device at low cost, we have used important primary Air quality sensors like MQ-135 and MQ-7 which are connected to the Arduino Pro Mini board. The MQ series of gas sensors uses a small heater inside with an electrochemical sensor. The output is an analog signal and can be read with an analog input of the Arduino [16]. The Arduino Pro Mini has the capability to handle eight analog data at a time, but in our experiment we have used only two sensors, thus only two sensing pins are used. To add the Bluetooth capability on AQMD, Bluetooth module HC-05 is used. Arduino Pro Mini has an interface between the sensor and the Bluetooth module. For communication between Arduino Pro Mini and Bluetooth module, Rx (receiving pin 0) and Tx (transmitting pin 1) are used. Whenever, there is a significant change in the air quality readings, Arduino Pro Mini communicates with Bluetooth module and then data are transmitted from AQMD to smart phone via Bluetooth. The prototype thus built is tested with the Arduino Uno board. The device weighs only around 69gm.

### 4.2 Mobile Application

The mobile application component of the AirSense is an Android application that is responsible for i) data collection from AQMD ii) forwarding data to the cloud service iii) providing AQImap iv) providing individual pollution footprint. The first task is to check the presence of a Bluetooth adapter. If the Bluetooth adapter is present, it checks if it is enabled. Smartphone users who are AQMD owners, enable it to volunteer in the AirSense platform.

Next, it checks for the paired/bonded devices, as the device must be paired before the application can use it. Here, the device IDs are being compared for a match. After getting the AQMD, a socket has to be created to handle the outgoing connection. Since the data from AQMD can be received at any point of time, so a thread is running to listen for incoming data. Geocoding API is used to get the location of the smartphone. For indoor, predefined location points are used. Next, to forward the data to the cloud, WiFiManager API of Android is used to manage all aspects of WiFi connectivity. However, 3G/4G connection may also be utilized instead. Mobile application provides local air quality data collected from the personal AQMD (or AQMDs in the vicinity). It also generates AQImap of the city from the collected data.

### 4.3 Cloud Service

In AirSense, OPENSHIFT [8], is utilized as a free cloud service provider (PAAS) that provides an environment to store, aggregate and analyze stored sensor data in the cloud server. Using the

REST API, an OPENSHIFT channel is created to store the data in the cloud. MySQL database is used for storing data in OPENSHIFT. The structure of the database is similar to Fig. 3. MAC\_ID of the devices is considered as the Primary Key for AirSense. Detailed specification of the database server and web server are given in Fig. 6. All the air quality parameters collected through the AirSense platform are aggregated for bigger picture. Air quality readings along with the locations are populated in the map to generate AQImap of the city, which can be visualized through the mobile application of the smartphone. An interaction diagram of the same is shown in Fig. 7.

It is evident from Fig. 7 that, there are some processing on both ends, i.e., in smartphone and also in the cloud server. In smartphone, both data cleaning and noise removal are done to get better accuracy of the sensed data.

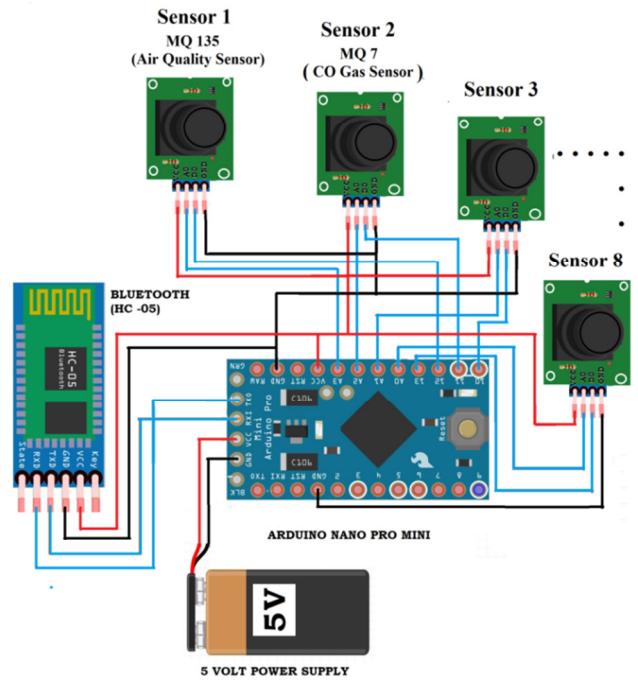


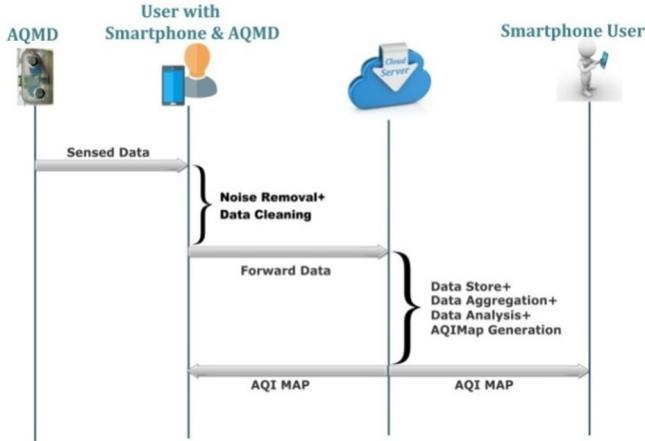
Figure 4. Circuit Diagram of the AQMD



Figure 5. AirSense in practical world



**Figure 6. Screenshot of OPENSHIFT cloud server for AirSense**



**Figure 7. Interaction diagram for AirSense**

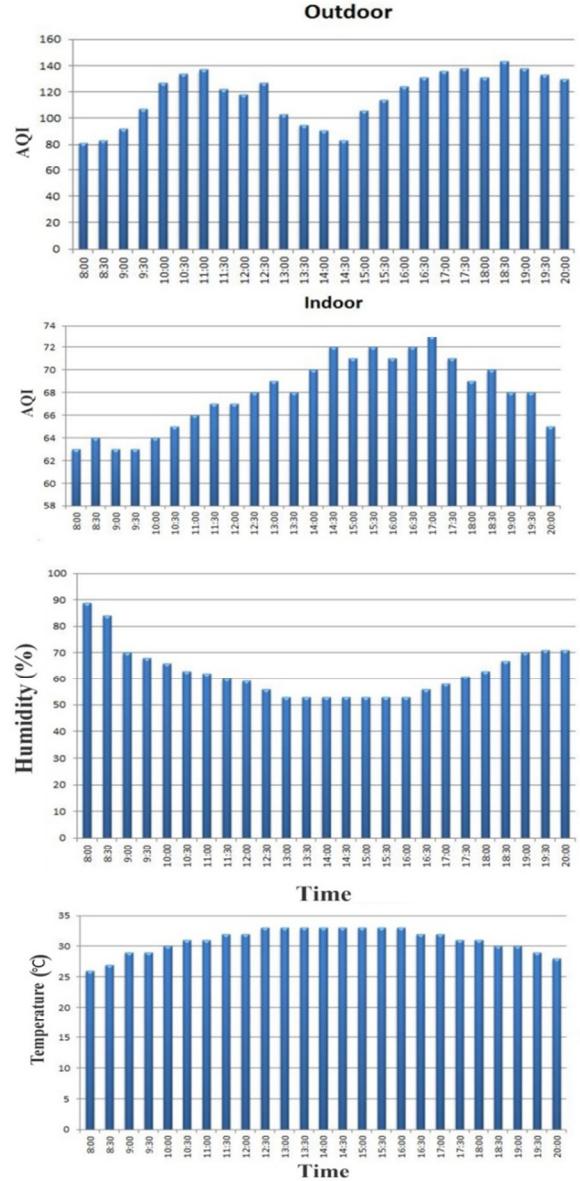
The data collected at the cloud server are processed. Data from different sensors like gas sensors, location are accumulated to get the individual pollution footprint. Data from different users are aggregated to form the AQIMap of a city. This can be used in smart cities as with only a few AQMD users around a city, citizens can get to know about AQIMap and hence can plan their route accordingly.

## 5. EXPERIMENTAL RESULTS

Experiments are conducted with 5 AQMD devices and Android based Smartphones from 5 users. For indoor scenario, data are collected from a floor of Department of Computer Science and Engineering of Jadavpur University. Outdoor data are collected from various places of the city and university campus as is evident from respective figures. In this paper two applications are discussed, AQIMap and individual pollution footprint. Before showing the output of each of these applications, we have studied the variation of AQI data with different time of day as shown in Fig. 8. Indoor data were taken around an air conditioned laboratory and does not show much variation as the factors influencing air quality does not vary much at the laboratory with air conditioning. However, outdoor environment as suspected, is much more susceptible to various transient factors that affect air quality throughout a day even at a particular location. We have shown all variations of temperature, humidity, Indoor AQI, Outdoor AQI with the change in time as shown in Fig. 8. It can be observed that, because of air conditioning, a stable and standard AQI reading can be obtained for indoor conditions, but it is interesting to find that the data behavior at outdoor also shows a pattern of more or less stable behavior for longer duration. During office hours, due to high traffic, air quality is quite poor, whereas

in the afternoon session the AQI level is moderate which actually signifies the traffic condition in turn.

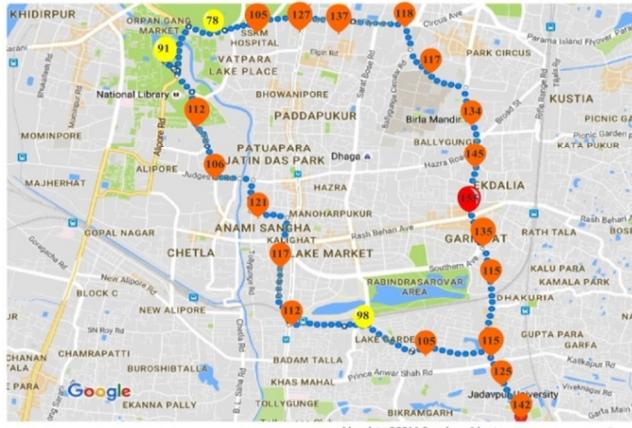
Micro level monitoring of air quality is possible with AirSense as an individual's pollution footprint in the city can be obtained for a day. Such a scenario is shown in Fig. 9. This helps an individual to monitor and hence prevent from sustained exposure to harmful pollutants. Macro level application of AirSense is generation of AQIMap for both indoor and outdoor scenarios. Fig. 10 shows the AQIMap generated for the outdoor and indoor environment. The outdoor area covered in our experiment is highly congested and traffic density is also high in the daytime. So, the outdoor environment is not found to be healthy for young and elder adults.



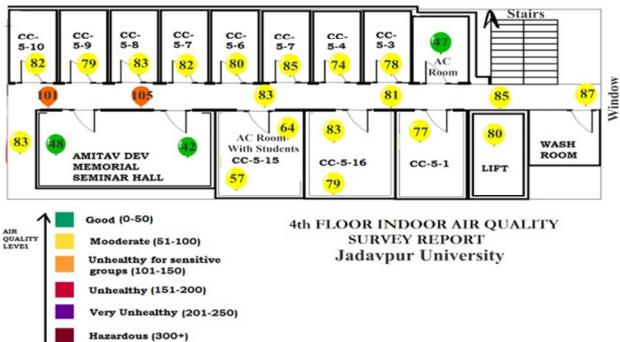
**Figure 8. Graph for Variation of AQI, Temperature and Humidity with Time of Day at a location both Indoor and Outdoor**

However, indoor conditions are better, as many of the rooms with air conditioning facility provides better air quality. Some part of the corridor shows worse quality due to the outlet of air conditioning units. This map can be used by the authority to

monitor and regulate air quality. This result can be validated with what we get in [17] and accepted worldwide as they use costly Air Quality Monitoring Devices. The advantage of AirSense is that, both macro and micro level air quality monitoring is possible. Reliable data is ensured as opportunistic crowd-sensing is used where a user cannot tamper with the data.



**Figure 9. Individual's (with AQMD) pollution footprint in Kolkata**



**Figure 10. AQImap for (a) outdoor and (b) indoor environment**

## 6. CONCLUSION

This paper presents the design of a simple and portable system, AirSense, that senses and collects air quality data thus enabling citizens to make informed decisions on managing and improving the environment. In the proposed opportunistic crowd-sensing inspired system, low cost, low power and light weight AQMDs with smartphones collaboratively perform sensing and data collection. Cloud is used to fulfil the requirements of computation and storage. Finally, a prototype is implemented and tested for both indoor and outdoor environments at macro and micro level. We plan to evaluate crowd sensing performance in more detail in future.

## ACKNOWLEDGMENT

The research work of the first author is supported by “Visvesvaraya PhD Scheme, Ministry of Communications& IT, Government of India”.

## REFERENCES

- [1] R. K. Ganti, F. Ye, and H. Lei, “Mobile Crowd-Sensing: Current State and Future Challenges,” IEEE Communication Magazine, vol. 49, no. 11, pp. 32-39, Nov. 2011.
- [2] Y. Zheng, F. Liu, and H. P. Hsieh, “U-Air: When Urban Air Quality inference meets Big Data,” in Proc. of ACM Conf. Knowledge discovery and data mining, Chicago, USA,pp. 1436-1444, Aug.2013.
- [3] D. Hasenfratz, O. Saukh, S. Sturzenegger, and L. Thiele, “Participatory Air Pollution Monitoring using Smartphones,” in Proc. of ACM Conf. Mobile Sensing, Beijing, China, April 2012.
- [4] A. Antonic, V. Bilas, M. Marjanovic, and M. Matijasevic, “Urban Crowd Sensing Demonstrator Sense the Zagreb Air,” in Proc. of IEEE Conf. Software, Telecommunications and Computer Network, Split, Croatia pp. 423-424, Sept. 2014.
- [5] C. Leonardi, A. Cappelotto, M. Caraviello, B. Lepri, and F. Antonelli, “SecondNose: An Air Quality Mobile Crowdsensing System,” in Proc. of ACM Conf. Human-Computer Interaction: Fun, Fast, Foundational, Helsinki, Finland, pp. 1051-1054, Oct.2014.
- [6] P. Dutta, P. M. Aoki, N. Kumar, A. Mainwaring, C. Myers, W. Willett, and A. Woodruff, “Demo Abstract: Common Sense-Participatory Urban Sensing using a Network of Handheld Air Quality Monitors,” in Proc. of the ACM Conf. Embedded Networked Sensor Systems, Berkeley, California pp. 349-350, Nov. 2009.
- [7] A. Antonić, M. Marjanović, K. Pripužić, and I.P. Zarko, “A Mobile Crowd Sensing Ecosystem Enabled by CUPUS: Cloud-based Publish/Subscribe Middleware for the Internet of Things,” Future Generation Computer Systems, vol. 56, pp. 607-622, March 2016.
- [8] Openshift: Red Hat’s Platform-as-a-Service (PaaS), URL:<https://www.openshift.com>
- [9] D. Oletic and V. Bilas, “Design of sensor node for air quality crowdsensing,” Sensors Applications Symposium (SAS), 2015 IEEE, Zadar, 2015, pp. 1-5.
- [10] N. Castell, M. Kobernus, H-Y. Liu, P. Schneider, W. Lahoz, A. J. Berre, J. Noll, Mobile technologies and services for environmental monitoring: The Citi-Sense-MOB approach, Urban Climate, Volume 14, Part 3, December 2015, Pages 370-382, ISSN 2212-0955
- [11] Aeroqual Air Quality Monitors, <http://www.aeroqual.com/indoor-air-quality-monitors/portable-monitors>, last accessed:10/06/2016
- [12] Aclima’s Air Quality Sensors for Streetview Cars, <https://9to5google.com/2015/07/28/aclimas-air-quality-sensors-are-being-attached-to-google-street-view-cars-in-san-francisco/>, last accessed: 10/06/2016
- [13] V. Sivaraman, J. Carapetta, K. Hu, B. G. Luxan, Hazewatch: A Participatory Sensor System for Monitoring Air Pollution in Sydney, Eight IEEE Workshop on Practical Issues in Building Sensor Network Applications 2013, pp. 56-64.
- [14] M. Mead, O. Popoola, G. Stewart, P. Landshoff, M. Calleja, M. Hayesb, J. Baldovi, M. McLeod, T. Hodgson, J. Dicks, A. Lewis, J. Cohen, R. Baron, J. Saffell, and R. Jones, “The Use of Electrochemical Sensors for Monitoring Urban Air Quality in Low-Cost, High-Density Networks,” Atmospheric Environment, vol. 70, pp. 186–203, May 2013.
- [15] “National Air Quality Index” by Central Pollution Control Board, Website: [cpcb.nic.in/FINAL-REPORT\\_AQI\\_.pdf](http://cpcb.nic.in/FINAL-REPORT_AQI_.pdf), last accessed: 14/10/2016
- [16] Arduino Compatible MQ gas sensors, Website: <http://playground.arduino.cc/Main/MQGasSensors>, last accessed: 16/10/2016
- [17] Air Pollution in the World, Realtime Air Quality Index; Website: <http://aqicn.org/search/#q=kolkata>; last accessed: 17/10/2016

# Energy Aware Cluster Head Load Balancing Scheme for Heterogeneous Wireless Ad Hoc Network

Rupam Some<sup>\*</sup>  
 Department of Information  
 Technology  
 Indian Institute of Engineering  
 Science and Technology  
 (Formerly BESUS)  
 rupam.some@hetc.ac.in

Tuhina Samanta  
 Department of Information  
 Technology  
 Indian Institute of Engineering  
 Science and Technology  
 (Formerly BESUS)  
 t\_samanta@it.iests.ac.in

Indrajit Banerjee  
 Department of Information  
 Technology  
 Indian Institute of Engineering  
 Science and Technology  
 (Formerly BESUS)  
 ibanerjee@it.iests.ac.in

## ABSTRACT

Wireless ad-hoc network is a resource constrained network applied in many applications. Clustering happened to be most effectual technique in order to attain steady performance and better utilization of energy. Traditional clustering approach elects a cluster coordinator so as to manage the whole cluster in terms of data collection, data fusion and transmission of the fused data to the distant base station. Random mobility in the phenomenal area leads to impose excess burden on the coordinator resulting quicker energy diminution which is analogous to minimum network life span. The proposed scheme exploits the heterogeneous sensor modules order to form the network and aims to balance the load of the cluster coordinator alias cluster head by concentrating on formation of nested cluster with the nodes imposing load so as to bypass the burden of the cluster head. The novelty of the scheme is not only to concentrate on load harmonization of the coordinator but also to establish an effectual arrangement of hustle free cluster formation and to prolong the network life time.

## CCS Concepts

- Networks → Network protocol design;

## Keywords

Load balancing; Nested cluster formation; Heterogenous mobile nodes.

## 1. INTRODUCTION

Load balancing in a wireless network improves the network stability by distributing the network load across the coordinators present in the network. Load balancing has

---

\*Mr. Some is the corresponding author.

several other positive consequences including prolonging the network lifetime, minimizing the energy dissipation, avoiding the collision in the network, controlling the congestion in the network. Recent research activities [1][2][3] concentrate on bypassing the load of the cluster head followed by data routing. The work is primarily focused on the establishment of associative cluster heads in order to sidestep the load followed by effectual data routing. On the other hand, traffic load distribution scheme proposed in [4-6] concentrates on balancing the load of the routing path followed by traffic load distribution.

In WLB-AODV[22], gateway load balancing scheme in internet integrated MANET has been proposed. In mobile ad hoc network (MANET) the frequent change in the large scale network topology leads to a challenge for delay sensitive data reception [8]. A most recent work in the similar domain namely Load Balanced Energy Efficient Clustering Protocol for Wireless Sensor Network [25] concentrated on virtual cluster formation followed by balancing the load of the CH in order to minimize the energy consumption of the network. However, the scheme aims to setup concentric virtual circles alias clusters putting the base station(BS) in the center of these clusters. In this algorithm, in each cluster, depending on the cluster size number of node is elected by the local information and also using the appropriate criteria which are called leader nodes which are responsible for collection and fusion of data from regular nodes within their leadership.

There are some important issues when conniving a load balancing system for mobile ad-hoc network(MANET). First, the load balancing scheme must be established within a short extent of time to acclimatize with the changes in the mobile nodes locations. Second, it should be instigated with smaller number of overheads in order to save energy. And last but not the least, the parameters that triggers the load balancing action must relate to the changes in the topology. In the present paper, we focus on harmonizing the load of the coordinator commonly known as a cluster head in cluster based mobile ad hoc network based of formation of nested cluster. The proposed scheme aims to achieve the objective by means of computing the load balancing factor (LBF) of the CH. Variation of the said factor triggers the load balancing scheme and starts establishment of the nested cluster with the nodes responsible for imposing the burden on the cluster head. On the contrary, nested cluster formation is bypassed if the number of load generating nodes are less than that

ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of a national government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only.

ICDCN '17 January 04-07, 2017, Hyderabad, India

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3018287>

of a predefined threshold thus initiating sleep schedule for the nodes resulting lesser energy depletion. The simulation results depicts the performance of the proposed scheme in terms of parameters like better utilization of energy, more network life time and number of alive nodes leading to a stable network for higher amount of time as compared to protocols in the similar family. Alongside that, energy efficiency in the said scheme is also attained through minimizing the energy required for cluster formation as compared to other recent activities in the said domain.

Some unique features of our proposed work are: (i) heterogeneous network environment (ii) mobile sensor nodes competent for any topological changes in the network. (iii) embedded clustering within network segment.

The rest of the paper is organized as follows: Section II reviews some of the approaches that deal with load balancing. Section III present the system model and energy model for the proposed system followed by the presenting the detailed description of the proposed system. The following section illustrates the performance of the system in terms of comparison with existing works. Section V draws the conclusion and present avenues for the future work.

## 2. RELATED WORK

Continued development of Micro electro Mechanical system (MEMS) and wireless communication technologies have empowered the development of large scale Ad Hoc networks. Due to limited power factor and inability to replenish the energy of the deployed sensors different innovative approaches were evolved to prolong the network life time [9]. Most of the approaches follow clustering approaches in order to minimize the depletion of energy and the communication load. In these conventional approaches, collected data is aggregated at the cluster coordinator followed by data fusion and the fused data is then transmitted to the base station. In cluster based communication scheme, the communication load of the cluster head (CH) is often high compared to rest of the cluster members [10]. Thus, harmonizing the load of the CH is essential for prolonging the network life time. LEACH [11] is a prevalent dynamic clustering technique in which after cluster formation, election of CH takes place and the role of CH is rotated among all the nodes in the cluster. The key snags faced in this approach lies in its inability to discard a node after it is being elected as a CH when the energy of the elected node lies beyond the threshold. Recent developments in the said field consequences in protocol alike Dynamic Load Balancing Protocol (DLBP) for Wireless Sensor Network [12] and An Energy Efficient Load Balancing Algorithm for Cluster Based Wireless Sensor Network (CELBA) [13]. DLBP exploits all the network nodes to achieve load balancing and enhance the life time of the network. This approaches to form a load balanced tree, eliminate the need for control messages at the time of data routing, keeps the load of the WSN balanced at the time of data routing, sends message to the next hop nodes without route discovery delay. DLBP provides a technique that keeps the load balancing of the WSN even after the sensors start collecting data from the environment and sending them to the sink. In DLBP, first phase of tree construction attempts to find at least one main path from any node to the sink. On the other hand, neighbor table is implemented to find alternative path to the sink. Every path to sink have

a cost and path having a lower cost is better to be used by a node. CELBA approaches to form clusters on the basis of load on the gateways. In this scheme the overall task is divided into phase such as discovery, balancing and clustering. The load balancing is achieved through calculating the communication cost by the gateway nodes with respect to every node in the cluster. This data is then sent to the sink node for further processing. The sink nodes perform the task of balancing the sensor nodes per gateway. Kuila et.al proposed an Energy Efficient Load Balanced Clustering Algorithm for Wireless Sensor Network (EELBC) [14]. EELBC is a mean heap based clustering algorithm in order to balance the load of the CH. The mean heap is built using the cluster heads on the no of sensor nodes allotted to the CHs. A ZigBee cluster based load balancing scheme [15] was proposed by Kartinah et.al where cluster heads exchange beacon frame with the neighbor CHs in order to have an idea of the cardinality of the neighbor CH. Thereafter, exchange of mobile nodes takes place amongst the CHs in order to balance the load of the neighbor. Load Spreading in Tree Type Routing Structure [8] was meant to balance the load of the routing path. In this approach maintenance of neighborhood metrics is presented as a means to preserve network objectives while achieving improved load distribution in a tree based routing pattern. An optimal load distribution scheme was proposed by chanak et.al [16] dealing with cluster head load balancing followed by efficient data routing. The protocol works on associative clustering model guided by residual energy and load of the sensor node in a cluster followed by avoidance of energy hole in order to support uniform load distribution. The load balancing phase is supported by associative cluster head which is established to bypass the load of the cluster head. In the recent progress in the field of WSN cluster formation and data transmission, Performance Analysis of Efficient Clustering Protocols for Maximizing the Lifetime of the Wireless Sensor network proposes two scheme named SEECP and MEECP for energy balanced cluster formation and data transmission respectively. In SEECP and MEECP (Single and Multi Hop Energy Efficient Clustering Scheme) [18] the cluster heads (CHs) are elected by a weighted probability based on the ratio between residual energy of each node and average energy of the network. The nodes with the high initial energy and residual energy will have more chances to be elected as CH. In MEECP, the elected CHs communicate the data packets to the base station via multihop communication approach. On the other hand, in Wireless Sensor Communication System Based on Direct Sum Source Coder [19] proposes a low delay as well as low complexity communication system for WSN bases on direct sum source coder. The source encoder converts a continuous amplitude signal into n tuple binary representation directly by a scalar quantization. The author derived the mean square error distortion of the said scheme from which the power allocation strategy can be optimally designed for each of the quantization bits. Recent evolution in the field of energy barring in the WSN reveals algorithm like Battery Optimal Scheduling Based On Energy Balance In Wireless Sensor Network[20] designed scheme which illustrates an scheduling scheme about which battery have to work based on remaining energy.

## 3. SYSTEM MODEL

Primary research in domain of wireless ad hoc networks

**Table 1: ZIGBEE PROFLEX01**

Operating Frequency	2.4 GHz
Line of Sight Range	1219 Meters
RX (receiving frame, -50 dBm)	149mA
Supply Range	2.0 V-3.3 V

**Table 2: ZIGBEE CC2520**

Operating Frequency	2.4 GHz
Line of Sight Range	440 Meters
RX (receiving frame, -50 dBm)	18.5 mA
Supply Range	1.8 V-3.8 V

concentrated on networks where all the nodes own identical software and hardware. This homogeneous construction of ad hoc network is attractive because it is resilient to individual failures. More recently, however, heterogeneous ad hoc networks have become prevalent, particularly in real life deployments because of their potential to increase the network lifetime and reliability without significantly increasing the cost. Typical heterogeneous ad hoc networks, involve a large number of low-priced nodes perform its assigned job alike sensing of the phenomena , whereas a few expensive nodes provide data fusion and transport. This segregation of tasks ensures a cost effective design as well as a more efficient implementation of the overall sensing application [21]. The primary objective of the work is to establish a ad hoc network model consists of N number of heterogeneous nodes deployed in a geographic region to continuously monitor an environment. The proposed heterogeneous network is set up of two types of nodes. The first type of node is high end in nature having higher transmission range and destined to be the cluster head upon deployment. On the other hand, the nodes of second category are designated as the member of the cluster. We denote the  $i^{th}$  sensor node by  $n_i$  and the corresponding set of nodes  $S = (n_1, n_2, \dots, n_n)$  where  $|S| = N$ . The proposed system model works as per the following assumptions:

- all the nodes are mobile
- Sensed data is transmitted to the cluster head (CH)
- CH aggregates the data and sends it to the base station (BS).
- All the nodes are position aware i.e. each node knows its location.
- The base station is unique and stationary with high energy

Zigbee modules, proFlex01 and CC2520, hold the nucleus of the proposed sensor network. Key specification of the Zigbee modules is depicted in Table I and Table II.

Compared to the modules involved in the traditional networks, the modules used in the proposed network is more effective in terms of excellent line of sight range. In terms of utility, the proposed network may be used for surveillance in the battle field for its high transmission range, high link budget and wide supply range. Considering the number of

CHs in the network be  $n_c$  and cardinality of a  $CH_i$  is  $X_i$ . We introduce the load balancing factor of the whole system so as to evaluate the load on the entire system as:

$$LBF_{system} = n_c / \sum_{i=1}^N (X_i - \mu)^2 \quad (1)$$

where  $\mu = n_c/(N - n_c)$  and  $n_c$ = total of nodes CH in the system. On the contrary, a slender change in the linear equation (1) leads to reckon the load imposed on an individual cluster head  $CH_i$  as:

$$LBF_{CH} = n_p / \sum_{i=1}^N (X_i - \mu)^2 \quad (2)$$

where  $\mu = n_p/(N - 1)$  and  $n_p$ = total number of sensor nodes present in the cluster.

## 4. ENERGY MODEL

In the proposed technique both the free space and multipath fading channel are used, depending on the distance between transmitter and the receiver. If the distance between transmitter and receiver is less than a threshold  $d_0$ , then free space model is opted; otherwise the multipath fading model is selected [25]. According to the model, energy required by the radio to transmit a  $k$  bit message through a distance  $d$  is depicted as below:

$$E(k, d) = k * E_{elec} + k * \varepsilon_{frespace} * d^2, d < d_0 \quad (3)$$

$$E(k, d) = k * E_{elec} + k * \varepsilon_{multipath} * d^4, d > d_0 \quad (4)$$

Where  $E_{elec}$  is the electronic energy needed by the circuit,  $\varepsilon_{frespace}$  and  $\varepsilon_{multipath}$  is the amplifier energy in case of selection of free space or multipath fading channel respectively. The radio also consumes energy to receive  $k$  bit message is depicted as:

$$E(k) = k * E_{elec} \quad (5)$$

### 4.1 Energy dissipation due to mobility of the sensor

Though, the proposed scheme considers that all the member nodes are mobile in nature,consideration of energy consumption due to physical movement of the mobile sensors is also incorporated in the energy model.If a sensor moves from  $(x, y)_{old}$  to  $(x, y)_{current}$ ,the displacement can be calculated as:

$$dis_k = \sqrt{(x_{old} - x_{current})^2 + (y_{old} - y_{current})^2} \quad (6)$$

For accurate calculation of energy dissipation due to movement of the sensors,energy dissipated due to mechanical movement and dissipation inside the motor due to factor like internal fraction was considered by Wang et.al[23],where the energy dissipation due to movement of a sensor  $k$  having angular velocity  $\omega_k$ ,acceleration  $\alpha_k$ ,deceleration  $\beta_k$  had been considered to compute the energy required to move to a distance of  $dis_k$  is:

$$E_{mobility} = \int_0^{f(\omega_k, \alpha_k, \beta_k, dis_k)} E(t) dt \quad (7)$$

Where  $E(t)$  is the power consumption.According to [25]  $E(t)$  can further be expressed as:

$$E(t) = V(t).I(t) \quad (8)$$

$$V(t) = RI(t) + K_b\omega(t) \quad (9)$$

$$I = 1/K_T [(J_{motor} + J_L d\omega/dt + T_L(t) + T_{fric} + D\omega(t)) \quad (10)$$

Where,  $E_t$  is the energy consumption.  $V_t$  is the voltage supplied to the motor and  $I_t$  is the current flow through the rotor of the motor.  $R$  is the armature resistance.  $K_b$ , is the back EMF constant.  $\omega(t)$  is the angular velocity of the motor at time  $t$ .  $K_T$  is the torque constant of the motor.  $J_L, J_{motor}$  is the inertia of load and motor.  $T_L(t)$  is the load torque at time  $t$ .  $T_{fric}$  is the friction torque of the motor.  $D$  is the viscous damping, which represents the coefficient of speed-dependent power dissipation.

## 5. PROPOSED SYSTEM

The prime target of the proposed technique is to design a mobile ad-hoc network having an efficient load balancing technique that certifies very low energy overhead and prolonged life time. The novelty of the proposed policy lies in its ability to balance the excess burden of the cluster head (CH) by forming a nested cluster inside the present cluster with the sensor nodes supposed to be accountable for imposing extra freight on the CH. The projected system achieves the objective of exploiting following phases:

1. Cluster formation
2. Runtime Load balancing

### 5.1 Cluster formation

After deployment of the sensors in the deployment area, the base station (BS) directs packet containing its location ( $x_{bs}, y_{bs}$ ) to all the nodes. Unlike other cluster head (CH) selection methods, this approach minimizes the burden of selection CH as high end nodes are destined to be CHs. After receiving the packet from the BS the CHs start sending a frame to all the sensor nodes within its transmission range. The frame encompasses the location of the CH ( $x_{ch}, y_{ch}$ ), location the BS ( $x_{BS}, y_{BS}$ ), residual energy of the CH ( $E_{residualCH}$ ) and CH ID ( $CH_{uid}$ ). Upon receiving the frame from the CH the nodes which are within the single hop distance ( $S_{single}$ ) from the CH sends an Acknowledge packet to the CH containing their location ( $x_{sh}, y_{sh}$ ) and residual energy ( $E_{residual}$ ) and node ID ( $S_{sID}$ ) to the CH. Nodes which are positioned in a multi hop distance with the CH retorts maintaining following steps:

1. Nodes ( $S_{multi}$ ) send a frame to the neighbor nodes within its range containing node ID ( $S_{nID}$ ), location ( $x_{mh}, y_{mh}$ ) and distance to the CH ( $D_{ch}$ ). The distance to the CH can be calculated as follows:

$$D_{CH} = \sqrt{(x_{mh} - x_{ch})^2 + (y_{mh} - y_{ch})^2}$$

2. Neighbor nodes replies the source node with a packet containing their location ( $x_{neighbour}, y_{neighbour}$ ), node ID ( $S_{neighbourID}$ ) and distance with the CH ( $D_{neighbour \rightarrow CH}$ ).
3. The source node  $S_{multi}$  calculates the distance with all the neighbours using:

$$D_{neighbour} = \sqrt{(x_{mh} - x_{neighbour})^2 + (y_{mh} - y_{neighbour})^2}$$

4. Finally, the source node N selects the neighbor with  $\text{Min}(D_{ch}$  and  $\text{Min}(D_{neighbour})$  and transmits an acknowledgement frame (ACK) comprises of node ID ( $S_{multiID}$ ), location ( $x_{mh}, y_{mh}$ ) and residual energy ( $E_{residual}$ ) to the CH through the nominated neighbor.

### 5.2 Runtime Load balancing

#### Definition:

*Cluster formation Threshold( $T_1$ )* : It is defined to be the minimum number of member nodes required to form the cluster. The said threshold  $T_1$  is extremely application specific and vary accordingly.

The proposed ad-hoc network consists of mobile sensor nodes. After accomplishment of every single round of cluster formation and data transmission, some sensor nodes may leave the current cluster region as well as some nodes from neighbor clusters may also take entry to the cluster. If the number of incoming nodes to the cluster is greater than that of the number of outgoing nodes from the cluster, a dense network network may be formed leading to routing congestion. Thus, the rudimentary target of the load balancing methodology is to lessen the burden of the CH in order to minimize the consumption of energy for the drive of maximizing the life time. The load balancing mechanism works following the course of action described below. As discussed in the preceding section the load balancing factor (LBF) of the entire system as well as of individual cluster can be premeditated with the help of equation (1) and (2). Initiation of the load balancing mechanism is centered on the succeeding conditions and they are depicted as follows:

1. If LBF of a  $CH_i(LBF_{CH})$  is equals to the LBF of the entire network ( $LBF_{system}$ ) then, it draws the conclusion of having a load balanced CH.
2. ) If LBF of a  $CH_i(LBF_{CH})$  is lesser than the LBF of the entire network ( $LBF_{system}$ ) then,  $CH_i$  is declared to have lesser load than that of the threshold.
3. ) If LBF of a  $CH_i(LBF_{CH})$  is greater than the LBF of the entire network ( $LBF_{system}$ ) then,  $CH_i$  is professed to have load greater than that of the threshold and as a consequence the proposed load balancing procedure is instigated.

To balance the burden of a CH, the proposed policy aims to form a nested cluster consisting of sensor nodes supposed to responsible for additional burden. To form a new nested cluster inside the present one, the CH compares the  $LBF_{CH}$  with the  $LBF_{system}$  iteratively for every node present in the cluster. When CH discovers the  $LBF_{CH}$  exceeds the  $LBF_{system}$  for a particular node  $n_i$ , the CH initiates to form a nested cluster entailing ( $X_{i-n_i}$ ) number of nodes, where  $X_i$  = cardinality of the CH. If the ( $X_{i-n_i}$ ) value is less than that of a threshold ( $T_1$ ) as required by the application then, CH instructs the additional number of nodes to switch off their radios to initiate a sleep schedule for prolonging the network life time. Details of the proposed algorithm is as follows:

### 5.3 Formation of nested cluster

Formation of the nested cluster is supported by calculation of density of mobile nodes within the area of activities. In order to, estimate the node density, the proposed scheme aims to divide the cluster into 4 segments supported by detection of location of the nodes. Immediately after the load balancing scheme is fired, the CH starts transmitting an advertise(ADV) message to the member nodes consisting the location of the CH ( $x_{ch}, y_{ch}$ ), requesting the members to notify their location and residual energy in order to denote their location within the segments specified. The member

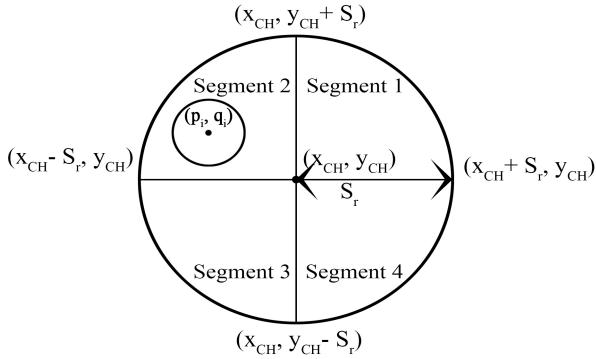
**Algorithm 1:** Balancing the load of the CH

**Input :**  $X_i$ (no of nodes in the cluster), $N$ (total no of nodes in the system,  
 $LIF_{CH}$ (Load indication factor of the CH),  
 $LIF_{system}$ (Load indication factor of the system),  
 $Y$ =load generated by individual node, $N$ =total number of nodes in the system;  
**Output:** Balanced load of the cluster head(CH)

```

1 begin
2   for i ← 1 to N do
3     Calculate:  $LIF_{CH}$  ,  $LIF_{system}$ 
4     if ( $LIF_{CH} > LIF_{system}$ ) then
5       calculate T= $LIF_{CH}-LIF_{system}$ ;
6       calculate Y= $LIF_{system}/N$ ;
7     end
8     for j ← 1 to  $x_i$  do
9       if ( $Y * (x_i)$ ) > T then
10        form nested cluster with  $x_i-i$  no of nodes;
11      else
12        initiate sleep schedule for ( $x_i-i$ ) no of nodes
13      end
14    end
15  end

```


**Figure 1: Segmented view of the cluster.**

nodes responses back with a reply(RPLY) message comprising their location to the CH. Upon receiving the locations, the CH starts implementing the following steps with the aim of calculating the node density.

- if the location of a node  $n_i$  is announced to be  $(p_i, q_i)$ , the node is said to be present in any one of the cluster segments depending on any one of the following inequalities to satisfy:

$$(x_{ch}-p_i)^2 + (y_{ch}-q_i)^2 - S_r^2 \leq 0, p_i > x_{ch}, q_i \geq y_i \quad (11)$$

$$(x_{ch}-p_i)^2 + (y_{ch}-q_i)^2 - S_r^2 \leq 0, p_i \leq x_{ch}, q_i > y_i \quad (12)$$

$$(x_{ch}-p_i)^2 + (y_{ch}-q_i)^2 - S_r^2 \leq 0, p_i < x_{ch}, q_i \leq y_i \quad (13)$$

$$(x_{ch}-p_i)^2 + (y_{ch}-q_i)^2 - S_r^2 \leq 0, p_i \geq x_{ch}, q_i < y_i \quad (14)$$

Where, $S_r$  is the transmission radius of the CH. Using any of the the equations (6), (7), (8), (9)signifies the node  $n_i$  to be present in

- CH evaluates the area of individual segments as :

$$\delta = \pi S_r^2 / 4 \quad (15)$$

- CH calculates the node density as per:

$$D_{segment} = P/\delta \quad (16)$$

where  $P$ =no of nodes present in the segment.

- CH aims to form the nested cluster in the segment having max( $D_{segment}$ ) node density and elects the node as the nested cluster head (NCH)in the specified segment having higher residual energy( $E_{residual}$ ).
- the NCH broadcasts a message to its neighbors requesting the location and residual energy of the neighbors( $E_{residual_{neighbor}}$ ) of the neighbors and calculates the distance to the neighbor( $d_i$ ).The NCH selects  $x_i-i$  number of member nodes (Algorithm 1) having min( $\sum_{i=0}^k d_i$ ) and max( $E_{residual_{neighbor}}$ )

**Algorithm 2:** Nested cluster formation

**Input :**  $(p_i, q_i)$ (location of  $i^{th}$  node),  
 $(x_{ch}, y_{ch})$ (location of the CH),  
 $S_r$ (transmission radius of the cluster),  
 $P$ (no of nodes in the segment),  
 $\delta$ (area of a segment),  
 $D_{segment}$ (node density within a segment),  
 $d_i$ (distance of a NCH with the  $i^{th}$ member ),  
 $E_{residual_{neighbor}}^i$ (Residual energy of  $i^{th}$  neighbor of NCH),  
 $E_{residual}^i$ (Residual energy  $i^{th}$  node in the cluster);  
**Output:** Detected cluster zone for nested cluster formation

```

1 begin
2   Calculate: $\delta = \pi S_r^2 / 4$   $\triangleright$  Area of a segment;
3   for i ← 1 to N do
4     Transmit: ADV message to receive  $(p_i, q_i)$  and  $E_{residual_{neighbor}}^i$ ;
5     Receive: RPLY message  $((p_i, q_i), E_{residual_{neighbor}}^i)$ ;
6     Evaluate: $(x_{ch}-p_i)^2 + (y_{ch}-q_i)^2 - S_r^2 \leq 0$ ;
7     if  $(p_i > x_{ch})$  and  $(q_i \geq y_{ch})$  then
8       |  $i^{th}$  node is in Segment1.
9     end
10    if  $(p_i \leq x_{ch})$  and  $(q_i > y_{ch})$  then
11      |  $i^{th}$  node is in Segment2.
12    end
13    if  $(p_i < x_{ch})$  and  $(q_i \leq y_{ch})$  then
14      |  $i^{th}$  node is in Segment3.
15    end
16    if  $(p_i \geq x_{ch})$  and  $(q_i < y_{ch})$  then
17      |  $i^{th}$  node is in Segment4.
18    end
19    Select NCH having max( $E_{residual}$ );
20    for i ← 1 to N do
21      Evaluate: $S = \sum_{i=1}^P d_i$  and  $E_{residual_{neighbor}}^i$ ;
22      if  $(\max(E_{residual_{neighbor}}^i))$  and  $\min(S))$  then
23        | Form the nested cluster with burdened nodes.
24      end
25    end
26  end
27 end

```

The time period for which a sleep scheduled sensor node has to switch off its radio is a function of time taken by the single hop and multi hop nodes to complete the transmission of sensed data to the CH. If the distance of  $i^{th}$  sensor node is  $d_i$  from the CH and the transmission range of the sensor

node is  $K_{tr}$ , the time taken by the sensor node to transmit a sensed phenomenon to the CH is formulated as:

$$T_{node \rightarrow CH} = \sum_{i=0}^N (d_i/S_r), 0 \leq d_i \leq K_{tr} \quad (17)$$

**Lemma 1:** Sleep time of the mobile node can be expressed as  $T_{node \rightarrow CH} = \sum_{i=1}^{q_1} (d_i/S_r) + \sum_{j=1}^{q_2} (d_j/S_r) + \dots + \sum_{k=1}^{q_n} (d_k/S_r)$ .

**Proof:** The proposed scheme ensures that nodes responsible for burdening the cluster head (CH) will craft a nested cluster inside the same cluster whose coordinator is overburdened. On the contrary, if the number of nodes responsible for generating the burden on the CH is less than that of a threshold, then formation of nested cluster will not be competent way of achieving energy efficiency thus, the CH instructs these nodes to initiate a sleep schedule. These nodes switch off their radio thus saving energy until communication between CH and rest of the nodes is over. Sleep time for the proposed scheme can be formulated ad as per equation (12).

More specifically, considering single and multi-hop nodes, the equation (12) can be expanded as:

$$T_{node \rightarrow CH} = \sum_{i=1}^{q_1} (d_i/S_r) + \sum_{j=1}^{q_2} (d_j/S_r) + \dots + \sum_{k=1}^{q_n} (d_k/S_r) \quad (18)$$

where,  $S_r$ =speed of propagation of data,  $q_1+q_2+\dots+q_n=n$ ,  $q_1, q_2, \dots, q_n$  are total no. of single hop, multihop and  $n$ -hop nodes respectively. The maximum sleep time period for which a sensor node has to switch off its radio can be expressed as:

$$\begin{aligned} T_{node \rightarrow CH}(\max) &= [(K_{tr} * q_1 + 2 * (K_{tr}) * q_2 + \\ &\dots + n * (K_{tr}) * q_n)]/S_r \quad (19) \end{aligned}$$

$$= (1/S_r) * \sum_{i=1}^{q_1} d_i + \sum_{j=1}^{q_2} d_j + \dots + \sum_{k=1}^{q_n} d_k$$

Where,  $S_r$ =speed of propagation of data,  $q_1+q_2+\dots+q_n=n$ ,  $q_1, q_2, \dots, q_n$  are total no. of single hop, multihop and  $n-th$  hop nodes respectively. The experimental result shows sleep time for a single hop and multi hop node is 19.684 microsecond and 31.2 microsecond respectively.

More precisely, if the coordinate of the CH, single hop, double hop and  $n^{th}$  hop node is  $(X_{ch}, Y_{ch})$ ,  $(a_i, b_i)$ ,  $(a_j, b_j)$  and  $(a_n, b_n)$  respectively we can compute  $d_i, d_j \dots d_n$  as:

$$\begin{aligned} d_i &= \sqrt{(x_{ch} - a_i)^2 + (y_{ch} - b_i)^2} \\ d_j &= \sqrt{(x_{ch} - a_i)^2 + (y_{ch} - b_i)^2} + \sqrt{(a_i - a_j)^2 + (b_i - b_j)^2} \\ d_n &= \sqrt{(a_n - a_j)^2 + (b_n - b_j)^2} + \sqrt{(x_{ch} - a_i)^2 + (y_{ch} - b_i)^2} + \\ &\quad \sqrt{(a_i - a_j)^2 + (b_i - b_j)^2} \quad (20) \end{aligned}$$

Through the aid of equation (15) sleep time period in equa-

**Table 3: Simulation Parameters**

Parameters	Value
Network Size	2000x2000m
Number of Sensor nodes	1000
Percentage of Cluster Head	5
Percentage of Mobile Nodes	100
Data Packet Size	4000bits
Sensing range(CH)	1219m
Sensing range(Member)	440m
Threshold value of Energy	0.1joule
Initial energy	2.0 joule

tion (14) can further be expanded as:

$$\begin{aligned} T_{node \rightarrow CH}(\max) &= 1/s_r * [\sum_{i=1}^{q_1} \sqrt{(x_{ch} - a_i)^2 + (y_{ch} - b_i)^2} + \\ &\sum_{j=1}^{q_2} \sqrt{(x_{ch} - a_i)^2 + (y_{ch} - b_i)^2} + \sqrt{(a_i - a_j)^2 + (b_i - b_j)^2} + \\ &\sum_{k=1}^{q_n} \sqrt{(a_n - a_j)^2 + (b_n - b_j)^2} + \sqrt{(x_{ch} - a_i)^2 + (y_{ch} - b_i)^2} + \\ &\sqrt{(a_i - a_j)^2 + (b_i - b_j)^2}] \quad (21) \end{aligned}$$

**Lemma 2:** The proposed method of load balancing has a time complexity of  $O(n^2)$ , where  $n$  is the number of nodes in the cluster.

**Proof:** Initially, the load balancing factor of the system ( $LBF_{system}$ ) and load balancing factor of the CH ( $LBF_{CH}$ ) is calculated and compared alongwith evaluation of amount of load imposed by individual node. This step requires iteration of  $n$  times resulting  $O(n)$  time. In the following steps the boundary condition for formation of the embedded cluster is executed by the repetition of  $n$  times resulting individual statement requiring  $O(n)$  time. Therefore, the aggregate time required by the statements in the inner iteration is  $O(n^2)$ . Henceforth, the algorithm requires overall run time of  $T(n)=O(1)+n+n+\dots+n=(n-1)/2=O(n^2)$ .

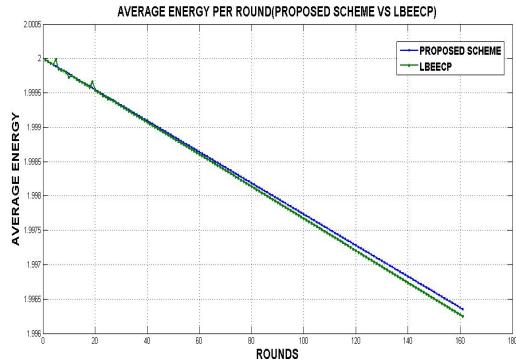
## 6. SIMULATION AND RESULT

### 6.1 Simulation Scenario

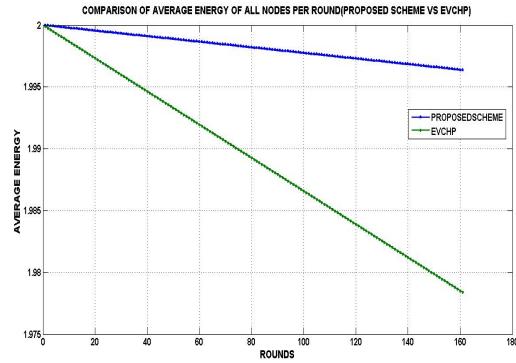
In order to evaluate the performance of the proposed arrangement, the protocol was simulated with the parameters shown in the table I.

Figure 2,3 and 4 illustrates the metric average energy of all nodes per round of the proposed scheme with respect to rounds. The simulation result depicts that the average energy of all deployed nodes per round is less than that of the conventional schemes, Load Balanced Energy Efficient Clustering Protocol for Wireless Sensor Network(LBEECP)[25], A Novel energy Efficient Vicce Cluster Head Routing Protocol in Wireless Sensor Network(EVCHP)[26] and Centralized Energy Efficient Load Balancing Algorithm (CELBA)[13].

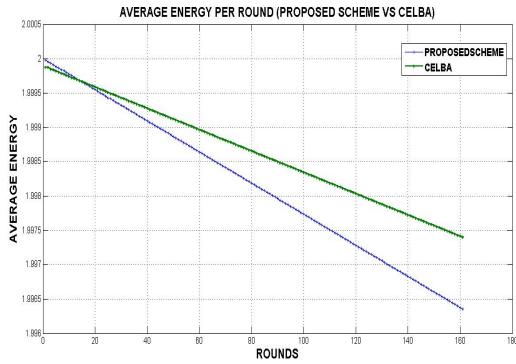
The proposed scheme achieves the average energy of 1.999 joule after all the rounds. In both the conventional arrangements[13][25][26], the overall process is substantial due to the elongated process of CH election and thereafter cluster formation followed by a load balancing scheme. In the proposed scheme, the process of CH election is very straightforward due to consideration of the high end node which are



**Figure 2:** Average energy of all nodes per round(Proposed scheme Vs LBEECP).

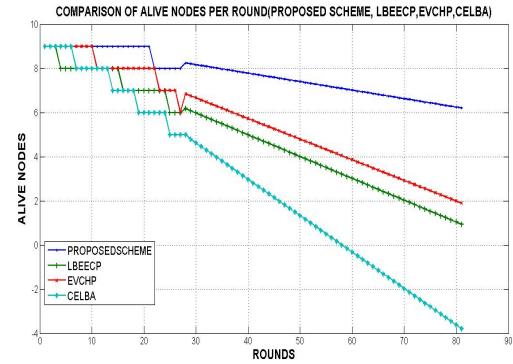


**Figure 3:** Average energy of all nodes per round(Proposed scheme Vs EVCHP).



**Figure 4:** Average Energy Per Round(Proposed Scheme Vs CELBA ).

predetermined to be elected as cluster heads, thus reducing the CH election phase. In addition to that, IN LBEECP due to the concentric cluster formation election of cluster head is not only computation heavy but also time consuming. The scheme choose the CH based on node degree standard second power average of neighbor nodes on special ray and also distance to BS. The node with high degree is elected as CH where as our scheme makes better utilization of resource in terms of less hazardous and straight forward process.



**Figure 5:** Alive nodes per round(Proposed Scheme,LBEECP,EVCHP,CELBA).

A firm performance can be attained only when the number of nodes deployed in the target area remains steady. In most of the recent works including LBEECP [25],EVCHP [26] and CELBA [13], due to a heavy process of cluster establishment and cluster coordinator election the rate of energy exhaustion is significantly high. Apart from that in EVCHP,in order to bypass the load,a common scheme of choosing an vice cluster head(VCH)so as to bypass the load of the CH,where VCH election again is computation heavy process and takes greater amount of time leading to faster energy depletion .

Figure 5 depicts, the proposed protocol, the number of nodes deployed remains same up to round 20, after that number of alive nodes decreases with a rate of 3.21 percentage. As conversed earlier, due to curtailed process of Cluster formation and CH election the amount of energy diminution is significantly less. Apart from that, in the said scheme load balancing is achieved by formation of nested cluster with the nodes responsible for generating excess burden on the CH and if the no of nodes generating burden is significantly less than that of a threshold then, instead of cluster formation the nodes initiate duty cycling until communication between rest of the nodes is over thus, leading to reduced energy exhaustion. On the contrary, EVCHP,the load of the CH is bypassed through the same common concept of declaring a leader node in order to handle the intra cluster commination as the size of the concentric clusters increases gradually.

The objective of presenting the projected scheme is to enhance the network life time in order to achieve better performance. To evaluate the network life time the simulation scenario was seasoned for 5 hours on a network of 1000 nodes. Figure 6 depicts the network lifetime of the proposed setup. It can noticed from Figure 6 that the nodes in the setup remains steady and consumes energy monotonically resulting steady network life time. The reason behind this perfection is making the whole process less computation heavy, initiation of sleep schedule for the nodes generating excess burden and curtailing the cluster formation and cluster head selection. Figure 7 illustrates the comparison of cluster formation energy required in the proposed scheme and Single hop Energy Efficient Clustering Protocol (SEECP)[20]. IN SEECP the cluster heads (CHs) are elected by a weighted probability based on the ratio between residual energy of each node and average energy of the network. The nodes with the high initial energy and residual

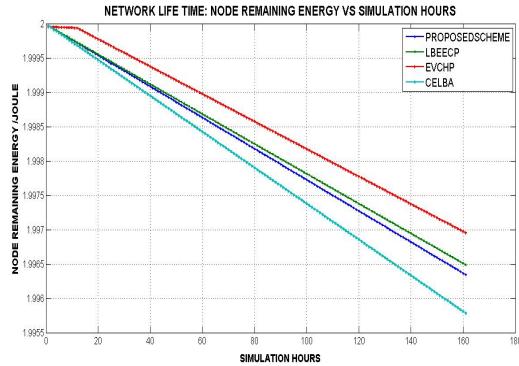


Figure 6: Network life time:Node remaining energy vs simulation hours.

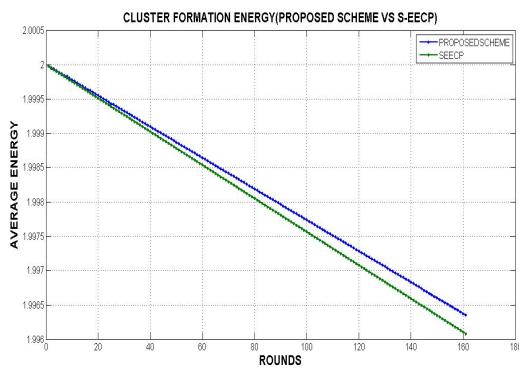


Figure 7: Cluster Formation energy (SEECP Vs proposed scheme).

energy will have more chances to be elected as CH. On the contrary, the proposed scheme works on a straight forward mechanism of cluster formation and CH election. Though the high end sensors are destined to play the role of CH, the cluster formation is concentrated around the deployed dedicated high end sensors. Pre-election of CH in said heterogeneous network not only gets rid of the computation heavy process but also saves energy. In the proposed scheme the average energy of all nodes in the cluster is 1.9999956J which is nearly equivalent to the energy with which the nodes were deployed , whereas , in SEECP, after cluster formation the average energy of a cluster is 1.870J which is significantly less from the energy with which they were deployed.

Figure 8 portrays the average energy of all the nodes in the first three rounds of simulation. Average energy of all nodes are 1.99983467, 1.999547063, 1.999341519, 1.999339519, and 1.999336119, 1.999333410 Joule respectively for round 1, 2 and 3. In the conventional protocols alike DLBP and CELBA, the first few rounds consumes most energy due to materialization of clusters, election of coordinators of the clusters and establishing communication between coordinator and members of the cluster. On the contrary, our proposed scheme gets rid of most of these phases via using heterogeneous ZigBee modules proflex01 and CC2520 where, proflex01 is predetermined to be coordinator thus it can form cluster based on its transmission radius resulting significantly lesser energy loss.

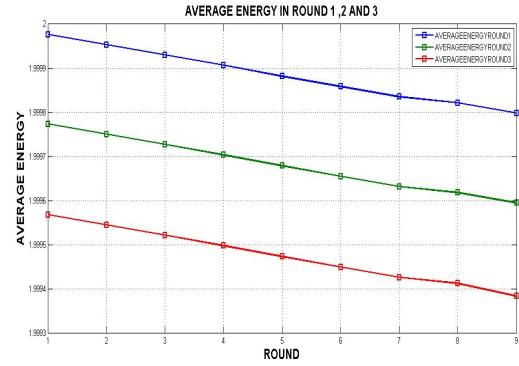


Figure 8: Average energy of all nodes in round 1,2,3.

## 7. CONCLUSION

Design of energy efficient load balancing technique so as to minimize the load of the cluster coordinator is a crucial concern for wireless ad-hoc network applications. In the proposed scheme we not only concentrated on balancing the load of the cluster head but also aims to improve the energy Saving arrangement. Implementation of the said scheme considers heterogeneous ZigBee modules having higher transmission capacity hence curtails the burden of the cluster construction and CH election. Alongside, the proposed scheme diminishes the depletion of energy by initiating a sleep schedule thus saving significant amount of energy.

## 8. REFERENCES

- [1] Dongmei Yan, Jinkuan Wang, Li Liu and Bin Wang, "Dynamic cluster formation algorithm target tracking-oriented," Computer Design and Applications (ICCDA), 2010 International Conference on, Qinhuangdao, 2010, pp. V4-357-V4-360.
- [2] S. Jin and K. Li, "LBBCS: A Load Balanced Clustering Scheme in Wireless Sensor Networks," Multimedia and Ubiquitous Engineering, 2009. MUE '09. Third International Conference on, Qingdao, 2009, pp. 221-225.
- [3] A. Thonklin and W. Suntiamorntut, "A load balanced cluster head election for uniform/non-uniform deployment over wireless sensor networks," Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on, Xi'an, 2011, pp. 488-492.
- [4] S. R. Deshmukh and V. T. Raisinghani, "EALBM: Energy aware load balancing multipath routing protocol for MANETs," Wireless and Optical Communications Networks (WOCN), 2014 Eleventh International Conference on, Vijayawada, 2014, pp. 1-7.
- [5] Yonghang Yan, Linlin Ci, Ruiping Zhang and Zhiming Wang, "Load balancing routing algorithm among multiple gateways in MANET with Internet connectivity," Advanced Communication Technology (ICACT), 2014 16th International Conference on, Pyeongchang, 2014, pp. 388-392.
- [6] Prasenjit Chanak, Indrajit Banerjee, Hafizur Rahaman, wireless sensor networks, Computers and Electrical Engineering, Volume 48, November 2015, Pages 343-357.
- [7] Fouzi Semchedine, Louiza Bouallouche-Medjkoune, Moussa Tamert, Farouk Mahfoud, Djamil Aïssani, "Load balancing mechanism for data-centric routing in wireless

- sensor networks”, Computers and Electrical Engineering, Volume 41, January 2015, Pages 395-406.
- [8] D. T. Delaney, L. Xu and G. M. P. O'Hare, ”Spreading the Load in a Tree Type Routing Structure,” Computer Communications and Networks (ICCCN), 2013 22nd International Conference on, Nassau, 2013, pp. 1-7.
- [9] B. Nazir and H. Hasbullah, ”Mobile Sink based Routing Protocol (MSRP) for Prolonging Network Lifetime in Clustered Wireless Sensor Network,” Computer Applications and Industrial Electronics (ICCAIE), 2010 International Conference on, Kuala Lumpur, 2010, pp. 624-629.
- [10] H. Aljawawdeh and I. Almomani, ”Dynamic Load Balancing Protocol (DLBP) for Wireless Sensor Networks,” Applied Electrical Engineering and Computing Technologies (AEECT), 2013 IEEE Jordan Conference on, Amman, 2013, pp. 1-6.
- [11] A. Tarachand, V. Kumar, A. Raj, A. Kumar and P. K. Jana, ”An Energy efficient Load Balancing Algorithm for cluster-based wireless sensor networks,” India Conference (INDICON), 2012 Annual IEEE, Kochi, 2012, pp. 1250-1254.
- [12] Pratyay Kuila, Prasanta K. Jana, Energy Efficient Load-Balanced Clustering Algorithm for Wireless Sensor Networks, Procedia Technology, Volume 6, 2012, Pages 771-777.
- [13] K. Zen, H. Lenando and M. N. Jambli, ”Load balancing based on nodes distribution in mobile sensor network,” Information Technology in Asia (CITA 11), 2011 7th International Conference on, Kuching, Sarawak, 2011, pp. 1-6.
- [14] D. T. Delaney, L. Xu and G. M. P. O'Hare, ”Spreading the Load in a Tree Type Routing Structure,” Computer Communications and Networks (ICCCN), 2013 22nd International Conference on, Nassau, 2013, pp. 1-7.
- [15] P. Chanak, T. Samanta and I. Banerjee, ”Cluster head load distribution scheme for wireless sensor networks,” SENSORS, 2013 IEEE, Baltimore, MD, 2013, pp. 1-4.
- [16] ] Q. Wang and T. Zhang, ”Traffic load distribution in large-scale and dense wireless sensor networks,” Wireless Internet Conference (WICON), 2010 The 5th Annual ICST, Singapore, 2010, pp. 1-8.
- [17] D. Kumar, ”Performance analysis of energy efficient clustering protocols for maximising lifetime of wireless sensor networks,” in IET Wireless Sensor Systems, vol. 4, no. 1, pp. 9-16, March 2014.
- [18] H. T. Nguyen, T. A. Ramstad and I. Balasingham, ”Wireless sensor communication system based on direct-sum source coder,” in IET Wireless Sensor Systems, vol. 1, no. 2, pp. 96-104, June 2011.
- [19] Rencheng Jin, Zhiping Che, Zhen Wang, Ming Zhu and Liding Wang, ”Battery optimal scheduling based on energy balance in wireless sensor networks,” in IET Wireless Sensor Systems, vol. 5, no. 6, pp. 277-282, 12 2015.
- [20] S. Kandukuri, N. Murad and R. Lorion, ”A single-hop clustering and energy efficient protocol for wireless sensor networks,” Radio and Antenna Days of the Indian Ocean (RADIO), 2015, Belle Mare, 2015, pp. 1-2.
- [21] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu and S. Singh, ”Exploiting heterogeneity in sensor networks,” Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies., 2005, pp. 878-890 vol. 2.
- [22] Rafi U. Zaman, Khaleel-Ur-Rahman Khan, and A. V. Reddy. 2010. Gateway load balancing in integrated internet-MANET using WLB-AODV. In Proceedings of the International Conference and Workshop on Emerging Trends in Technology (ICWET '10). ACM, New York, NY, USA, 411-416.
- [23] Guiling Wang, M. J. Irwin, P. Berman, Haoying Fu and T. La Porta, ”Optimizing sensor movement planning for energy efficiency,” ISLPED '05. Proceedings of the 2005 International Symposium on Low Power Electronics and Design, 2005., 2005, pp. 215-220.
- [24] ElectreCrttft Corporation. DC Motors, Speed Controls,Servo Systems, An Engineering Handbook. Pergamon Press, 1977.
- [25] S. Siavoshi, Y. S. Kavian and H. Sharif, ”Load-balanced energy efficient clustering protocol for wireless sensor networks,” in IET Wireless Sensor Systems, vol. 6, no. 3, pp. 67-73, 6 2016.
- [26] T. A. H. Hassan, G. Selim and R. Sadek, ”A novel energy efficient vice Cluster Head routing protocol in Wireless Sensor Networks,” 2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS), Cairo, 2015, pp. 313-320.

# Practical Multi-threaded Graph Coloring Algorithms for Shared Memory Architecture

Nandini Singhal, Sathya Peri, Subrahmanyam Kalyanasundaram

Department of Computer Science & Engineering

Indian Institute of Technology Hyderabad

{cs15mtech01004, sathya\_p, subruk}@iith.ac.in

## ABSTRACT

In this paper, we present multi-threaded algorithms for graph coloring suitable to the shared memory programming model. Initially, we describe shared memory implementations to the algorithms widely known in the literature like Jones Plassman graph coloring. Later, we propose new approaches to solve the problem of coloring using mutex locks while making sure that deadlocks do not occur. Using datasets from real world graphs, we evaluate the performance of all these algorithms on the Intel platform. We compare the performance of sequential graph coloring v/s our proposed approaches and analyze the speedup obtained against the existing algorithms from the literature. The results show that the speedup obtained by our proposed algorithms in terms of the time taken for coloring is consequential. We also provide a direction for future work towards improving the performance further in terms of different metrics.

## CCS Concepts

- Computing methodologies → Shared memory algorithms; Concurrent algorithms; Distributed algorithms;

## Keywords

Graph coloring; multi-threaded; shared memory; locks; barrier

## 1. INTRODUCTION

The Graph Coloring Problem pertains with attributing colors to the vertices of a simple graph such that no two adjacent vertices get the same color (also termed as vertex coloring). Proper coloring of an arbitrary graph using number of colors equal to its chromatic number is known to be an NP-hard problem. Thus, the primary goal in the graph coloring problem is to reduce the coloring time and minimize the number of colors used (ensuring proper coloring).

With the growing use of multi-core systems, hardware capability can be completely exploited with parallel algorithms. Each core can independently process a subtask and this can speed up the overall performance of the algorithm.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

*ICDCN '17, January 04-07, 2017, Hyderabad, India*

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3018281>

However, sequential algorithms act only on single core, albeit availability of multi-cores. The development of multi-core systems has been followed by the advent of shared memory programming paradigms like OpenMP and Pthreads, which are very easier to program. In this paper, we simulate using Pthreads for fine grained control over thread management.

The basic motivation behind this paper is to limit the number of threads being created at runtime (irrespective of the size of the graph to be colored). It is a very common practice in the literature dealing with parallel graph algorithms to create a thread corresponding to each vertex in the input graph. This is not practically feasible owing to the constraints on the resources (stack size, etc.) available. Also, increasing the number of threads beyond the hardware capacity does not lead to any improvement in the performance of the algorithm. To highlight this point, we have evaluated a parallel algorithm for increasing number of threads as shown in Figure 1. We see that with continuous increase in number of threads, the performance starts worsening. In this paper, we present algorithms for graph coloring which facilitate the user to input the number of threads to be acted upon depending on the hardware architecture (hardware threads, number of cores) available.

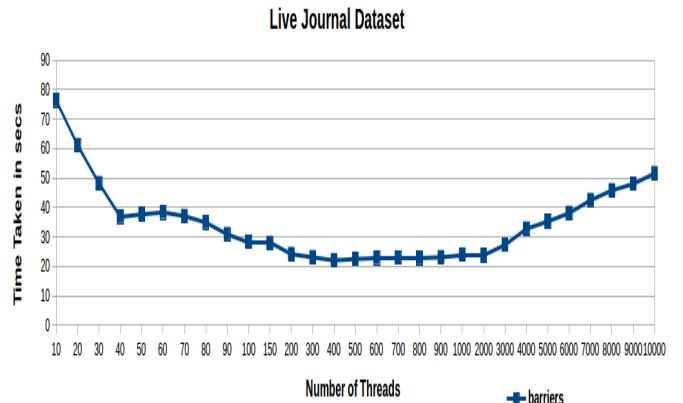


Figure 1: Time Taken in secs v/s Number of Threads

We look at four different approaches: (1) barrier synchronization; (2) jones plassman; (3) locking variants and (4) transactions. The first algorithm deals with the most widely studied technique of coloring using barrier synchronization [1, 2, 5]. A barrier is a command in the source code for a set of threads/processes which acts as a synchronization point for all of them. When the threads execute the barrier point, they must stop and cannot proceed until all remaining

threads/processes have reached the barrier. The literature about parallel graph coloring refers mainly to the algorithm using barrier synchronization. However, they do not prove the correctness of the algorithm. This is crucial because all the threads run in an asynchronous manner and the behaviour of a thread at a particular time instant cannot be determined. In this paper, we present a modified version of this algorithm in section 3 and give a proof of correctness of the algorithm. Subsequently, we also present a shared memory implementation of the Jones Plassman Algorithm. The later subsections puts forth new approaches for graph coloring using some standard locking techniques. We ensure that deadlocks do not occur. In Section 4, we then evaluate the performance of the presented algorithms against the sequential greedy coloring algorithm. We see that the performance of our proposed approach (using locks) outweighs the other existing algorithms on shared memory architecture. Finally, section 5 concludes and provides a direction of future work in this area.

## 2. BACKGROUND

### 2.1 Problem Definition

A graph  $G$  is represented as a pair  $(V, E)$  of a set of vertices  $V$  and a set of edges  $E$ . The edges are unordered pairs of the form  $\{i, j\}$  where  $i, j \in V$ . Two vertices  $i$  and  $j$  are said to be adjacent if and only if  $\{i, j\} \in E$  and non-adjacent otherwise. The degree of a vertex  $v$  is the number of vertices adjacent to  $v$  and is denoted by  $\deg(v)$ .

#### The Graph Coloring Problem:

A vertex coloring of a simple graph is an allotment of colors to the vertices such that no two adjacent vertices are assigned the same color. It is easy to see that any arbitrary simple graph can be colored with  $\Delta + 1$  colors where  $\Delta$  is the maximum degree of the simple graph.

### 2.2 Related Work

The problem of parallel graph coloring has been studied extensively, even on multi and many-core architectures [1–4]. Jones & Plassman [8] proposed a distributed graph coloring algorithm in which they process the vertices of a graph in a random order. The difficulty with this approach lies in identifying the most effective ordering of the vertices according to the graph in question. Hasenplaugh, et al. [6] proposed two ordering schemes which performs efficiently on the shared memory implementation of the Jones Plassman Algorithm. Gebremedhin & Manne [5] presented a basic parallel graph coloring algorithm using block partitioning. However, it can be seen that it is highly inefficient owing to the synchronization in each iteration of tentative coloring phase. Also, all the vertices with conflicting colors are colored sequentially. This leads to reduced parallelism in the algorithm where number of conflicts are high. Gebremedhin, Manne & Woods [4] propose several enhancement over [5] to reduce the number of conflicts by the use of graph partitioning packages. However, since the underlying algorithm itself does not completely exploit the parallelism, this algorithm does not fare too well.

Boman, et al. [1] presented a novel distributed graph coloring algorithm based on the previous notions, by improving the parallelism. This algorithm also assumed that the number of

processors available would be significantly less as compared to the number of vertices in the graph. Hence it partitioned the input graph and assigned a subset of vertices to each processor. Çatalyürek, et al. [2] extend the previous notion for shared memory model. However, the algorithm create threads for each vertex in the input graph in each iteration of the algorithm. This increases the overhead significantly. Also, the paper does not provide a proof as to why the algorithm would terminate in a finite number of steps or would result in proper coloring of the graph eventually (without the use of locks for accessing shared memory).

### 2.3 System Model

In this paper, we assume that our system consists of  $n$  processors, accessed by  $p$  threads/processors that run in a completely asynchronous manner. Hence, we make no assumption about the relative speeds of the processors. We also assume that none of these processors and threads fails.

## 3. SOLUTION APPROACHES

In this section, we present various approaches suitable for dealing with the Graph Coloring problem on a shared memory programming model which means that threads communicate only by writing to and reading from the shared memory. We begin by assuming that all the vertices in the graph are assigned unique id's from  $\{1, 2, \dots, |V|\}$ . Initially, the graph  $G = (V, E)$  has been preprocessed by partitioning it uniformly into  $p$  partitions where  $p$  is the number of threads. Then the vertices with their corresponding id's in  $\{1, \dots, |V|/p\}$  are assigned to partition  $V_1$ ,  $\{|V|/p + 1, \dots, 2|V|/p\}$  get assigned to partition  $V_2$  and so on until  $V_p$ . It seems only fair that the graph partitioning preprocessing time be included in the overall time taken for graph coloring. Hence, using this random heuristic based partitioning helps us bring down the overall time taken for coloring. Therefore, we do not use a graph partitioning software for minimizing the crossing edges between various partitions because it effectively increases the coloring time.

The vertices in each partition/block can be classified into: internal vertices (whose all the neighbouring vertices lie in the same partition) and boundary vertices (those who have neighbours belonging to other partitions). Each thread is responsible for proper coloring of all the vertices in its partition. The subsequent subsections present algorithms using the First Fit Coloring strategy, which assigns each vertex the least legal color available. All the algorithms can be easily adapted to other coloring strategies like Largest Degree First, etc.

### 3.1 Using Barrier Synchronization

The barrier synchronization based algorithm has been widely explored in the literature [1, 2, 5]. However, the major difference in the algorithm presented here is that [2] has not explicitly used barriers for synchronization. We use barriers because it is more efficient than creating new threads as in [2]. As discussed before, a barrier is a synchronization point amongst threads. This algorithm has two phases: tentative coloring and conflict detection phase. Each thread maintains a copy of colors assigned to its neighbours locally in *ForbiddenColors* List. In the first phase, a thread assigns a color to all the vertices in its partition by taking into account all its previously colored neighbours from the local copy. However, it might result in two threads simul-

taneously coloring the vertices adjacent to each other with the same color. Hence, in the second phase, each thread  $T_i$  checks whether the vertices in  $V_i$  have been assigned valid colors by comparing the color of each vertex against all its neighbours. If any vertex and its neighbor have the same color, then the vertex in the partition with lower partition id is marked for recoloring.

The first and second phases are synchronized by a barrier that ensures that all the  $p$  threads start their execution at the same instant. This is crucial because if a thread were still coloring while other tries to detect conflicts, then this can lead to false detection eventually leading to improper coloring. The algorithm has been described in Algorithm 1.

---

**Algorithm 1** Using Barrier

---

```

1: Input:  $p \leftarrow$  no of threads
2: uniform partitioning of  $V$  into  $V_1, V_2, \dots, V_p$  in increasing order
   of vertex ids
3:  $m \leftarrow$  maximum degree of graph
4: procedure PARALLELGRAPHCOLORING( $G = (V, E)$ )
5:   for all thread  $T_i | i \in \{1, \dots, p\}$  do
6:     Identify boundary vertices of partition  $i$ 
7:     Initialise  $TotalColors[m + 1] \leftarrow \{0, 1, \dots, m\}$ 
8:     for  $v \in V_i$  do
9:       Create List  $v.ForbiddenColors$ 
10:      Initialise  $v.ForbiddenColors$  to  $-1$ 
11:    end for
12:     $U_i \leftarrow V_i$ 
13:    while  $U_i \neq \emptyset$  do
14:      for each  $v \in U_i$  do            $\triangleright$  Phase 1 starts
15:        Assign  $color(v) \leftarrow \min\{TotalColors - v.ForbiddenColors\}$ 
16:        for each  $u \in \text{adjacent}(v) | u \in V_i$  do
17:          Update color( $v$ ) in  $u.ForbiddenColors$ 
18:        end for
19:      end for
20:      Wait for all threads to reach here     $\triangleright$  Using barrier
21:       $R_i \leftarrow \emptyset$                     $\triangleright$  Phase 2 starts
22:      for each  $v \in U_i | v$  is a boundary vertex in  $U_i$  do
23:        for each  $u \in \text{adjacent}(v) | u \notin V_i$  do
24:          Update color( $u$ ) in  $v.ForbiddenColors$ 
25:          if  $color(u) = color(v) | u \in V_j$  and  $i < j$  then
26:             $R_i \leftarrow R_i \cup \{u\}$ 
27:          end if
28:        end for
29:      end for
30:       $U_i \leftarrow R_i$ 
31:      Wait for all threads to reach here     $\triangleright$  Using barrier
32:    end while
33:  end for
34: end procedure

```

---

**Lemma 1:** *Barrier Synchronization Algorithm results in proper coloring of the graph.*

**Proof:** Let us prove by contradiction. So, we assume that the barrier synchronization algorithm does not result in proper coloring of the graph meaning that two adjacent vertices in the graph have the same color at the end of the algorithm.

Each round/iteration of the algorithm consists of 2 phases: coloring and conflict detection phase respectively. We denote the coloring phase of  $i^{th}$  iteration as  $i.1$  and conflict detection phase of  $i^{th}$  iteration as  $i.2$ .

Without loss of generality, let us say that a vertex  $v_x$  which is adjacent to a vertex  $v_y$ , gets colored  $c$  in round  $i.1$  and vertex  $v_y$  gets assigned the same color in round  $j.1$ , both belonging to different partitions and  $i \leq j$ .

$$color^{i.1}(v_x) = color^{j.1}(v_y) = c \text{ where } v_x, v_y \text{ belong to different partitions}$$

Now there are two possibilities as follows:

a)  $v_x$  was assigned color  $c$  in round  $(j - 1).1$ : In this case, in round  $(j - 1).2$ ,  $v_x$  and  $v_y$  would be identified with same color and the vertex in the lower partition id would get recolored in round  $j.1$ . Hence either  $v_x$  or  $v_y$  would have a color different from  $c$ . Also since  $i \leq j$ , this means that both vertices get properly colored. Hence this is a contradiction to our initial assumption.

b)  $v_x$  was assigned a color different from  $c$  in round  $(j - 1).1$ : In this case,  $v_x$  got recolored back to color  $c$  in round  $j.1$ , then a conflict will be detected in round  $j.2$  and it will be resolved in round  $(j + 1).1$ . Hence it again contradicts our assumption.

Thus we can conclude that eventually all the conflicts get resolved and no two adjacent vertices get assigned to a same color.  $\square$

**Lemma 2:** *Barrier Algorithm terminates after a maximum of  $p + 1$  iterations.*

**Proof:** The partitions of the graph are  $V_1, V_2, \dots, V_p$ . In each round, a vertex in  $V_i$  is recolored if it has a conflict with a vertex in  $V_{i+1}, \dots, V_p$ .

In the  $1^{st}$  round, at least all vertices of  $V_p$  get properly colored and all conflicts of the vertices in  $V_{p-1}$  with vertices of  $V_p$  are identified, which are resolved in the next round. Similarly, in the  $2^{nd}$  round, all vertices of  $V_{p-1}$  get properly colored and all conflicts in  $V_{p-2}$  with  $V_{p-1}$  are identified, which are resolved in the subsequent round.

Thus by induction, it is easy to see that after  $p + 1$  iterations,  $V_1$  gets properly colored. Also, the maximum number of times a vertex in partition  $V_i$  gets recolored is  $(p - i)$ .  $\square$

## 3.2 Jones Plassman Algorithm

The Jones Plassman algorithm is a very popularly known distributed graph coloring algorithm. In this subsection, we present a shared memory implementation of the Jones Plassman Algorithm. Initially, each vertex  $v$  in the input graph is assigned a distinct random number  $\rho(v)$ . This is helpful in ordering the vertex amongst its neighbours by computing local data structures,  $n\text{-wait}(v)$  and  $send-list(v)$ . The data structure  $n\text{-wait}(v)$  maintains the count of those neighbours  $u \in adj(v)$  which have  $\rho(u)$  greater than  $\rho(v)$ . This implies that vertex  $v$  should be colored after coloring of these vertices. Similarly,  $send-list(v)$  keeps those neighbours  $u \in adj(v)$  which have  $\rho(u)$  smaller than  $\rho(v)$ , meaning that once vertex  $v$  gets colored, the vertices in  $send-list(v)$  can be colored.

---

**Algorithm 2** Jones Plassman

---

```

1: Input:  $p \leftarrow$  no of threads
2: Assign  $\rho(v) \forall v \in V$                                  $\triangleright$  distinct, random number
3: procedure PARALLELGRAPHCOLORING( $G = (V, E)$ )
4:   for all thread  $T_i | i \in \{1, \dots, p\}$  do
5:     Identify boundary vertices of partition  $i$ 
6:     Initialise  $TotalColors[m + 1] \leftarrow \{0, 1, \dots, m\}$ 
7:      $color\_list \leftarrow \emptyset$ 
8:     Initialise Concurrent List  $L_i$                        $\triangleright$  indicating all adjacent
   vertices of all vertices in  $T_i$ 

```

---

Since the vertices of different partitions in the graph communicate by exchanging messages in the distributed algorithm, we achieve the same in the shared memory by using a concurrent list data structure. For each thread  $T_i$ , a corresponding Concurrent Set based List  $L_i$  is initialised. The

algorithm proceeds in iterations until all the vertices in partition local to each thread get colored. In each iteration, all the vertices with their  $n\text{-wait}$  as 0, say  $P$ , are colored indicating their turn in the ordering amongst the set of neighbours. As a result of this, the vertices which have been waiting for  $P$  to get colored ( $\text{send-list}(P)$ ) have to be informed. So,  $P$  is added to the Concurrent List of the corresponding threads (those partition which contains vertices in  $\text{send-list}(P)$ ).

---

```

9:   for each  $v \in V_i$  |  $v$  is a boundary vertex do
10:     $n\text{-wait}(v) \leftarrow 0$ 
11:     $\text{send-list}(v) \leftarrow \emptyset$ 
12:    for each  $u \in \text{adjacent}(v)$  |  $u$  is a boundary vertex do
13:      if  $\rho(u) > \rho(v)$  then
14:         $n\text{-wait}(v) \leftarrow n\text{-wait}(v) + 1$ 
15:      else
16:         $\text{send-list}(v) \leftarrow \text{send-list}(v) \cup \{u\}$ 
17:      end if
18:    end for
19:    if  $n\text{-wait}(v) = 0$  then
20:       $\text{color-list} \leftarrow \text{color-list} \cup \{v\}$ 
21:    end if
22:  end for
23:  for  $v \in V_i$  do
24:    Create List  $v.\text{ForbiddenColors}$ 
25:    Initialise  $v.\text{ForbiddenColors}$  to  $-1$ 
26:  end for
27:  Invoke Color( $\text{color-list}$ )
28:  Invoke Append_Concurrent_List( $\text{color-list}$ )
29:   $n\text{-colored} \leftarrow |\text{color-list}|$ 
30:   $\text{color-list} \leftarrow \emptyset$ 
31:  while  $n\text{-colored} <$  no of boundary vertices in  $V_i$  do
32:    Iterate  $L_i$  to get  $v$ 
33:    for each  $u \in \text{adjacent}(v)$  |  $u \in V_i$  and  $u$  is a boundary
vertex do
34:       $n\text{-wait}(u) \leftarrow n\text{-wait}(u) - 1$ 
35:      Update color( $v$ ) in  $u.\text{ForbiddenColors}$ 
36:      if  $n\text{-wait}(u) = 0$  then
37:         $\text{color-list} \leftarrow \text{color-list} \cup \{u\}$ 
38:      end if
39:    end for
40:    Invoke Color( $\text{color-list}$ )
41:    Invoke Append_Concurrent_List( $\text{color-list}$ )
42:     $n\text{-colored} \leftarrow n\text{-colored} + |\text{color-list}|$ 
43:     $\text{color-list} \leftarrow \emptyset$ 
44:  end while
45:  for each  $v \in V_i$  |  $v$  is a internal vertex in  $V_i$  do
46:     $\text{color}(v) \leftarrow \min\{\text{TotalColors} - \text{color}(\text{adjacent}(v))\}$ 
47:  end for
48: end for
49: end procedure
50: procedure APPEND_CONCURRENT_LIST( $\text{color-list}$ )
51:  for each  $v \in \text{color-list}$  do
52:     $\text{temp\_set} \leftarrow \emptyset$ 
53:    for each  $w \in \text{send-list}(v)$  do
54:       $\text{temp\_set} \leftarrow \text{temp\_set} \cup \text{Partition\_id}(w)$ 
55:    end for
56:    for each  $k \in \text{temp\_set}$  do
57:       $L_k.\text{insert}(v, \text{color}(v))$ 
58:    end for
59:  end for
60: end procedure
61: procedure COLOR( $\text{color-list}$ )
62:  for each  $v \in \text{color-list}$  do
63:     $\text{color}(v) \leftarrow \min\{\text{TotalColors} - v.\text{ForbiddenColors}\}$ 
64:    for each  $u \in \text{adjacent}(v)$  do
65:      Update color( $v$ ) in  $u.\text{ForbiddenColors}$ 
66:    end for
67:  end for
68: end procedure

```

---

A thread keeps iterating through its concurrent set  $L_i$  to check if any other vertex (which it has been waiting on) has been colored. If a new insertion happens in the List then the corresponding adjacent vertex's  $n\text{-wait}$  is decreased by 1. It is to be noted that the Concurrent Set based Linked List with functions for scan and append can be implemented

using mutexes or atomic CAS operations. Here, delete operation is not required. Also, the scan function simply checks if the next pointer is not NULL. If not NULL, it indicates a new element has been inserted in the List. The complete algorithm is described in Algorithm 2.

**Lemma 3:** *At least one vertex at every instant in JP Algorithm has its  $n\text{-wait}$  as 0.*  $\square$

**Lemma 4:** *Jones Plassman Algorithm results in proper coloring of the graph.*

**Proof:** This can be seen in a straightforward manner. Improper coloring can only happen if two adjacent vertices of different partitions are colored at the same time by different threads wherein they both read the same colors of the neighbours and assign the adjacent vertices to the same color. Now, as can be seen from Algorithm 2, only vertices with their  $n\text{-wait}$  as 0 can be colored at a particular time instant. Thus, it is to be shown that no two adjacent vertices have their  $n\text{-wait}$  as 0 at the same time instant. It is known that all vertices in the graph are assigned a distinct, random number. Hence because of lines 15-18 in Algorithm 2, all adjacent vertices would be waiting on one another. This proves that  $n\text{-wait}$  of two adjacent vertices cannot become 0 at the same instant and thus ensures legal coloring.  $\square$

**Lemma 5:** *Jones Plassman Algorithm terminates after a finite iterations.*

**Proof:** In case of a complete graph of  $n$  vertices, each vertex in a partition sends a message to each of the concurrent thread. Say a vertex in the 1st partition is currently having  $n\text{-wait}$  as 0. Hence it colors itself and writes to Concurrent List of all other  $p-1$  threads. Now all vertices decrease their  $n\text{-wait}$  by 1. At least one vertex now has its  $n\text{-wait}$  to be 0. Hence it again colors itself and sends messages to all other  $p-1$  threads. Since each partition contains at max  $n/p$  vertices, the maximum number of messages passed by a single thread =  $(p-1) * n/p = O(n)$ . Now since there are  $p$  threads, the total number of messages are upper bounded by  $O(n * p)$ .  $\square$

### 3.3 Using Mutex Locks

The motivation behind using an alternative to the barrier synchronization approach presented in previous subsection lies in the fact that, since all the threads get synchronized at two points (Lines 20 & 31 in Algorithm 1) in each iteration, there is an unfavourable impact on the performance of the algorithm. The overall goal is to avoid global synchronization of threads and let them run independently. We present algorithms based on the locking of the graph vertices. With locks, coloring the vertex becomes a critical section and a thread can only enter the critical section when it has acquired the lock.

#### 3.3.1 Coarse Grained Locking

With Coarse Grained Locking, there exists a big lock on the complete list of boundary vertices. This implies that at any point, a thread must acquire a lock on this list to color any boundary vertex.

#### 3.3.2 Fine Grained Locking

Coarse Grained Locking can be improvised on by making use of fine grained locks wherein each vertex has a corresponding lock. A thread wishing to color a boundary vertex has to obtain locks on all the neighboring vertices of that

boundary vertex. However, to avoid deadlock, a global ordering of vertices is maintained (based on their vertex ids) and vertices acquire locks in the respective order. The complete algorithm has been described in Algorithm 3.

### Algorithm 3 Using Fine Grained Locks

```

1: Input:  $p \leftarrow$  no of threads
2: procedure PARALLELGRAPHCOLORING( $G = (V, E)$ )
3:   for all thread  $T_i \mid i \in \{1, \dots, p\}$  do
4:     Identify boundary vertices in  $V_i$ 
5:     Initialise  $TotalColors[m + 1] = \{0, 1, \dots, m\}$ 
6:     for each  $v \in V_i \mid v$  is a internal vertex in  $V_i$  do
7:        $color(v) \leftarrow \min\{TotalColors - color(adjacent(v))\}$ 
8:     end for
9:     for each  $v \in V_i \mid v$  is a boundary vertex in  $V_i$  do
10:      List  $A_i \leftarrow adj(v) \mid adj(v)$  is a boundary vertex
11:       $A_i \leftarrow A_i \cup \{v\}$ 
12:      Lock all vertices in  $A_i$  in increasing order of vertex ids
13:       $color(v) \leftarrow \min\{TotalColors - color(adjacent(v))\}$ 
14:      Unlock all vertices in  $A_i$ 
15:    end for
16:  end for
17: end procedure

```

#### 3.3.3 Cutting Waiting Chains

The drawback of using fine grained locks as described in the previous subsubsection is that in case of a chain graph, each vertex could be waiting on its adjacent vertex to acquire the locks. These waiting chains can worsen in denser graphs. This leads to a motivation to develop an algorithm for cutting long transitive waiting chains. Here, we present a variant of the Anderson, et al. [7] algorithm.

The idea here is to maintain a table data structure of boolean fields with number of rows equal to the number of partitions + 1 of the graph and number of columns equal to the number of partitions. A request for coloring a vertex  $v$  can be made by positioning itself in the column indexed by  $partition\_id(v)$ , of a particular row. Each true entry present in a row in the table corresponds to the request positioned in that row. Each column is indexed by the partition id's of the graph. The advantage to using this approach is that the size of the table is very small (in order of number of partitions) as compared to the size of input graph. The requests are fulfilled in the increasing order of rows starting from the first row of the table. To maintain validity, we need to ensure that at any instant only one request can be placed by a particular thread in some row.

To place a request for coloring a vertex  $v$  of a particular partition in a particular row, the thread needs to check the existing requests placed in the corresponding row. If all vertices corresponding to the requests placed in that row are not adjacent to  $v$  in the graph, then the request can be placed in the corresponding row. For this, a thread must know about the vertices which are being colored corresponding to the partition id's whose entry is true in the respective row. To achieve this, an atomic array of the size of the number of partitions is maintained. Each element of the array indexed by a partition id corresponds to the vertex being colored from that partition. This information is updated whenever a new request is placed.

The terminology used in the pseudo code is consistent with [7]. The shared memory field  $head$  is an atomic field which points to the currently enabled row in the table. Two more atomic arrays  $enabled$  and  $numReq$  are maintained. Each row in the table has a corresponding entry in these

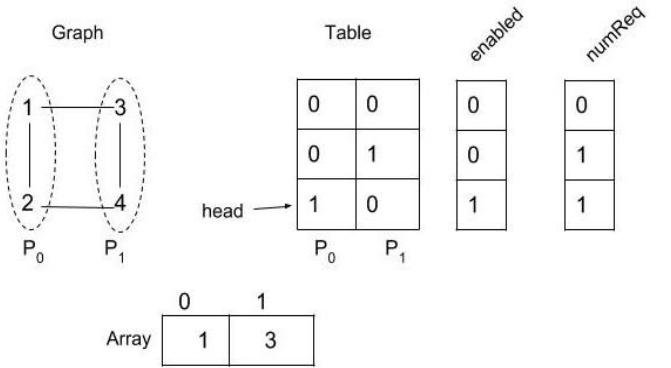


Figure 2: Illustration of working of variant of Anderson's algorithm

arrays indexed by the row number. A true entry in the  $enabled$  array indicates that the requests positioned in the corresponding row are fulfilled. At a particular instant, only one row can be enabled.  $numReq$  array maintains the count of the number of requests in each row. The last request in a row to be fulfilled, marks the next row to be *enabled*. Locks are needed to avoid concurrent changes to a row. Hence the algorithm proceeds in hand-over-hand row wise locking. This prevents deadlocks and allows fine grained concurrent access over the table data structure. Figure 2 illustrates the working of the algorithm when vertices 1 and 3 of partitions  $P_0$  and  $P_1$  are being colored concurrently. This can be identified by atomic  $VertexFromPartition$  array. It can be seen that the request for vertex 1 is placed in first row and vertex 2 in second row respectively. The first row of the table is referenced by  $head$  which corresponds to *enabled* being set to true. The number of requests placed in a given row is stored in  $numReq$ . Algorithm 4 describes the complete pseudo-code of the algorithm.

### Algorithm 4 Anderson's Variant

```

1: Input:  $p \leftarrow$  no of threads
2:  $A \leftarrow$  Atomic array of  $p$  indices
3: Initialise a table data structure of boolean fields with #rows =  $p + 1$ , #cols =  $p$ 
4: procedure PARALLELGRAPHCOLORING( $G = (V, E)$ )
5:   for all thread  $T_i \mid i \in \{1, \dots, p\}$  do
6:     Identify boundary vertices in  $V_i$ 
7:     Initialise  $TotalColors[m + 1] = \{0, 1, \dots, m\}$ 
8:     for each  $v \in V_i \mid v$  is a internal vertex in  $V_i$  do
9:        $color(v) \leftarrow \min\{TotalColors - color(adjacent(v))\}$ 
10:      end for
11:      for each  $v \in V_i \mid v$  is a boundary vertex in  $V_i$  do
12:        Invoke  $request\_table(v, i)$ 
13:         $color(v) \leftarrow \min\{TotalColors - color(adjacent(v))\}$ 
14:        Invoke  $release\_table(v, i)$ 
15:      end for
16:    end for
17:  end procedure
18: procedure REQUEST_TABLE( $v, i$ )
19:   Update  $A[i] \leftarrow v$ 
20:   Acquire a lock on the  $head$  row of the table
21:    $start \leftarrow head$ 
22:   while true do
23:     Initialise List  $L \leftarrow \emptyset$ 
24:     for each true entry  $\in table[start][k]$  do where  $1 \leq k \leq p$ 
25:       List  $L \leftarrow A[k] \cup L$ 
26:     end for
27:      $flag \leftarrow false$ 
28:     for each  $w \in L$  do
29:       if  $w$  is adjacent to  $v$  then

```

---

```

30:      flag  $\leftarrow$  true
31:      end if
32:    end for
33:    if flag = false then
34:      Goto line 43
35:    end if
36:    next  $\leftarrow$  start + 1
37:    Acquire a lock on the next row of the table
38:    Release lock on the start row of the table
39:    start  $\leftarrow$  next
40:  end while
41:  Set table[start][i]  $\leftarrow$  true
42:  numReq[start]++
43:  Release lock on the start row of the table
44:  Wait until enabled[start] is set to true
45: end procedure
46: procedure RELEASE_TABLE(v, i)
47:   Acquire a lock on the start row of the table
48:   Set table[start][i]  $\leftarrow$  false
49:   numReq[start]--
50:   if numReq[start] = 0 then
51:     next  $\leftarrow$  start + 1
52:     enabled[start]  $\leftarrow$  false
53:     head  $\leftarrow$  next
54:     enabled[next]  $\leftarrow$  true
55:   end if
56:   Release lock on the start row of the table
57: end procedure

```

---

At any instant, only one request for coloring a vertex can be placed from each partition in this concurrent data structure. In the case of a complete graph, each request would be placed in a different row. Also since all the requests are fulfilled in the increasing order of the rows, there is no request that cannot be placed on its arrival. Hence the algorithm terminates when all the requests have been enabled. It can be seen that in case of a chain graph over  $n$  vertices, the length of the waiting chain equals to 2 only. Every alternate vertex can be placed in the same row of the table. This is an improvement over fine grained locking where the length of the waiting chain could have been  $n$ .

### 3.3.4 Maximal Independent Sets of subgraphs

It is commonly known that computing a Maximal Independent set of a graph is an NP-Complete problem. In this subsection, we present an algorithm for graph coloring which maintains small subgraph of the original graph and computes the MIS on it. The keypoint is that at any instant, the maximum of vertices in the subgraph is equal to the number of partitions. Since the size of the subgraph is very small as compared to the input graph; the algorithm is expected to fare well practically. Whenever a request for coloring a vertex  $v$  is made by a thread, a node is added to the subgraph  $G'$  corresponding to the thread's partition. If  $v$  is adjacent to any vertex in  $G$ , then the corresponding edge is added to vertex's partition node in  $G'$ , if present. The algorithm proceeds in iterations until all the vertices in its partition get colored. In each step, an MIS is identified and those vertices are marked as *active*, which means that they can be colored. Furthermore, after a vertex has been colored, it is removed from the subgraph along with its corresponding edges. To avoid concurrent accesses to the shared subgraph  $G'$ , a coarse mutex lock is used. The pseudo code is described in Algorithm 5.

## 3.4 Using Transactions

A transaction is a piece of code which executes atomically. Since the internal vertices are colored without any interaction amongst threads, they can be colored without creating any transaction. However, each boundary vertex has to be

---

### Algorithm 5 MIS in subgraphs

---

```

1: Input:  $p \leftarrow$  no of threads
2: uniform random partitioning of  $V$  in  $V_1, V_2, \dots, V_p$ 
3: Initialise an empty graph data structure  $G'$ 
4: procedure PARALLELGRAPHCOLORING( $G = (V, E)$ )
5:   for all thread  $T_i \mid i \in \{1, \dots, p\}$  do
6:     Identify boundary vertices in  $V_i$ 
7:     Initialise  $TotalColors[m + 1] = \{0, 1, \dots, m\}$ 
8:     for each  $v \in V_i \mid v$  is a internal vertex in  $V_i$  do
9:       color( $v$ )  $\leftarrow$  min{ $TotalColors - color(\text{adjacent}(v))$ }
10:    end for
11:    for each  $v \in V_i \mid v$  is a boundary vertex in  $V_i$  do
12:      Invoke  $request\_graph(v, i)$ 
13:      color( $v$ )  $\leftarrow$  min{ $TotalColors - color(\text{adjacent}(v))$ }
14:      Invoke  $release\_graph(v, i)$ 
15:    end for
16:  end for
17: end procedure
18: procedure REQUEST_GRAPH( $v, i$ )
19:   Acquire a lock on the graph  $G'$ 
20:   Add a vertex  $i$  to  $G'$  and mark it inactive
21:   Add edges from  $i$  to respective vertices  $\in G'$  and vice versa
22:   if degree of  $i = 0$  then
23:     Mark  $i$  as active
24:   end if
25:   Release lock on graph  $G'$ 
26:   Wait until vertex  $i$  becomes active
27: end procedure
28: procedure RELEASE_GRAPH( $v, i$ )
29:   Acquire a lock on the graph  $G'$ 
30:   Remove vertex  $i$  from  $G'$  and all corresponding edges
31:   Identify MIS from the set of inactive vertices and mark active
32:   Release lock on  $G'$ 
33: end procedure

```

---

colored by creating a transaction. A *read* operation is performed for reading the colors of all the adjacent vertices and finally a *write* is invoked for assigning a valid color to the boundary vertex. If at any point, an operation fails, the transaction has to be restarted. Once the transaction commits, this implies that the vertex has been colored (by writing its color to shared memory graph). This algorithm ensures proper coloring of the graph. We simulate this using Basic Timestamp Ordering (BTO) Protocol. This has been described in Algorithm 6.

## 4. SIMULATION RESULTS & ANALYSIS

We performed our tests on 24 core Intel Xeon server (X5675) running at 3.07 GHz core frequency. Each core supports 6 hardware threads, clocked at 1600 MHz. In the experiments conducted, the time taken for coloring the graph in the multi-threaded version includes the time taken for partitioning of graph as well. However, time taken for coloring in all versions (sequential & parallel) excludes the time taken to read the graph input. Each data point is obtained after averaging for 10 iterations. To test the performance of the algorithms, we have used real world graph, Live Journal from SNAP [9]. We have evaluated for two metrics: Time Taken to color the graph and Number of Colors Used. We have tested it for all algorithms in previous section by varying the number of threads in the range 1-1000 and noted it for the best result.

As can be clearly observed from the performance results, the barrier synchronization and Jones Plassman Algorithm do not fare well and are not comparable to the sequential coloring. On the other hand, locks and transactions seem to perform fairly well in terms of time taken for coloring maintaining a reasonable number of colors used. We observe that fine grained locking performs significantly better

---

**Algorithm 6** Using Transactions

---

```

1: Input:  $p \leftarrow$  no of threads
2: Declare color( $v$ )  $\forall v \in V$  in shared memory
3:  $aborts \leftarrow 0$ 
4: Initialise Protocol (BTO/SGT)
5: procedure PARALLELGRAPHCOLORING( $G = (V, E)$ )
6:   for all thread  $T_i | i \in \{1, \dots, p\}$  do
7:     Identify boundary vertices in  $V_i$ 
8:     Initialise  $TotalColors[m + 1] = \{0, 1, \dots, m\}$ 
9:     for each  $v \in V_i | v$  is a internal vertex in  $V_i$  do
10:       color( $v$ )  $\leftarrow \min\{TotalColors - \text{color}(\text{adjacent}(v))\}$ 
11:     end for
12:     for each  $v \in V_i | v$  is a boundary vertex in  $V_i$  do
13:       Begin Transaction
14:       List  $C \leftarrow$  read color( $\text{adj}(v)$ )
15:       if read fails then
16:         Abort transaction & goto line 15
17:       end if
18:       write color( $v$ )  $\leftarrow \min\{TotalColors - C\}$ 
19:       try_commit() transaction
20:       if try_commit() fails then
21:         Increment aborts & goto line 15
22:       end if
23:     end for
24:   end for
25: end procedure

```

---

as compared to the sequential coloring. It is important to realise that Jones Plassman Algorithm does not fare well in terms of the time taken for coloring, partly because of the inefficient random ordering assigned to vertices. Hence even though it uses a reasonable number of colors, it is not practically feasible.

In the literature, there exists many ordering schemes which can further reduce the number of colors used such as Largest Degree First, Saturation Degree, etc [10]. It is important to note that since these orderings require some sorting of the vertices to order them, these will incur additional cost in terms of time taken. For parallel graph coloring, such heuristics can be used to order the vertices of each partition. Hence if such techniques are employed, they will lead to a rise in the time taken proportionally amongst all the algorithms (including the greedy sequential one). Thus, there will be no impact on the relative performance of the algorithms employed for a different ordering heuristic.

Table 1: Results of Live Journal Dataset

Algorithm	#threads	Time Taken (secs)	#colors used
Fine Grained Locks	70	6.18	334
Transactions (BTO)	200	8.26	335
Sequential algorithm	1	13.86	334
Coarse grained locks	100	17.75	333
Maximal Independent Set	2	18.36	336
Anderson's variant	14	19.26	335
Barrier synchronization	400	21.99	334
Jones Plassman	40	64954	334

## 5. CONCLUSION & FUTURE WORK

We have presented parallel algorithms for graph coloring suitable to the shared memory programming model. We have looked into the most commonly used approach for coloring using barrier synchronization and Jones Plassman Algorithm. We have also proposed new approaches using locks.

Using the SNAP dataset, we evaluated the performance of the algorithms on the Intel platform. The results show that the improvement is noteworthy. This gives a motivation that the overhead of locking and unlocking operations is less and they do scale well with increasing number of threads as compared to the existing approaches.

We intend to test these algorithms for other types of graphs including dense ones. It seems that the algorithm can be improved by exploring pushing ahead of requests in the table used in Anderson's algorithm. Also these locking ideas can be extended for specific categories of graphs such as trees, star, etc. Furthermore, cutting the waiting chains caused by fine grained locking in graphs is an active research problem.

## 6. REFERENCES

- [1] Erik G Boman, Doruk Bozda\u011f, Umit Catalyurek, Assefaw H Gebremedhin, and Fredrik Manne. A scalable parallel graph coloring algorithm for distributed memory computers. In *Euro-Par 2005 Parallel Processing*, pages 241–251. Springer, 2005.
- [2] \u011fmit V. \u011falyurek, John Feo, Assefaw Hadish Gebremedhin, Mahantesh Halappanavar, and Alex Pothen. Graph coloring algorithms for multi-core and massively multithreaded architectures. *Parallel Computing*, 38(10-11):576–594, 2012.
- [3] Mehmet Deveci, Erik G. Boman, Karen D. Devine, and Sivasankaran Rajamanickam. Parallel graph coloring for manycore architectures. In *2016 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2016, Chicago, IL, USA, May 23-27, 2016*, pages 892–901, 2016.
- [4] Assefaw H Gebremedhin, Fredrik Manne, and Tom Woods. Speeding up parallel graph coloring. In *Applied Parallel Computing. State of the Art in Scientific Computing*, pages 1079–1088. Springer, 2006.
- [5] Assefaw Hadish Gebremedhin and Fredrik Manne. Scalable parallel graph coloring algorithms. *Concurrency - Practice and Experience*, 12(12):1131–1146, 2000.
- [6] William Hasenplaugh, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson. Ordering heuristics for parallel graph coloring. In *26th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '14, Prague, Czech Republic - June 23 - 25, 2014*, pages 166–177, 2014.
- [7] Catherine E. Jarrett, Bryan C. Ward, and James H. Anderson. A contention-sensitive fine-grained locking protocol for multiprocessor real-time systems. In *Proceedings of the 23rd International Conference on Real Time and Networks Systems, RTNS 2015, Lille, France, November 4-6, 2015*, pages 3–12, 2015.
- [8] Mark T. Jones and Paul E. Plassmann. A parallel graph coloring heuristic. *SIAM J. Scientific Computing*, 14(3):654–669, 1993.
- [9] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. <http://snap.stanford.edu/data>, June 2014.
- [10] Md Mostafa Ali Patwary, Assefaw H Gebremedhin, and Alex Pothen. New multithreaded ordering and coloring algorithms for multicore architectures. In *Euro-Par 2011 Parallel Processing*, pages 250–262. Springer, 2011.

# Balancing Maximal Independent Sets using Hadoop

Ashish Bhuker  
 Indian Institute of Technology  
 Patna,Bihar  
 ashish.cs12@iitp.ac.in

Rajiv Misra  
 Indian Institute of Technology  
 Patna,Bihar  
 rajivm@iitp.ac.in

Bhanu Pratap Singh  
 Indian Institute of Technology  
 Patna,Bihar  
 bhanu.pcs15@iitp.ac.in

## ABSTRACT

Given a graph  $G = (V, E)$ , an independent set  $I$  is a subset of vertices, such that no two vertex in  $I$  share a common edge. A maximal independent set (MIS) is an independent set such that there is no way we can add any other vertex such that the property of independent set remains intact. Different MIS heuristics are used in parallel applications for identifying the subset of independent tasks. Traditional MIS heuristics aim to generate minimum number of MIS i.e. number of parallel steps for the particular applications. However, if MIS produced have skewed sizes, hardware resource utilization becomes inefficient. Parallel computing has become a key feature for improving performance in these days. A task is broken down into various subtasks and each subtask is executed in parallel on various processors. This is where generating and balancing the maximal independent sets come into play. In this paper, in the context of parallel application, we have proposed a heuristic for balancing the maximal independent sets using Hadoop after dividing the graph into various maximal independent sets. We have verified through experiment that our heuristic balances MIS without much increasing the number of MIS generated.

## CCS Concepts

• Computing methodologies → *Parallel computing methodologies; MapReduce algorithms; Parallel algorithms;*

## Keywords

Parallel Computing, Balanced MIS, Hadoop

## 1. INTRODUCTION

Using multiple processors and combining their output to solve a single problem is called parallel programming. The main focus in parallel programming is to decide how to decompose the problems into sub-problems that can be execute simultaneously and how a sub-problem will relate with other

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [Permissions@acm.org](mailto:Permissions@acm.org).

*ICDCN '17 January 04-07, 2017, Hyderabad, India*  
 ©2017 ACM. ISBN 978-1-4503-4839-31701 ...\$15.00  
 DOI: <http://dx.doi.org/10.1145/3007748.3018282>

sub-problems. In parallel computing, scheduling of identified sub-problems onto the processing units of a platform is an important step. In such situations, one would need to maximize the amount of parallel execution attained in a given step as well as need to minimize the total number of steps needed. In such cases, where the computational dependency between entities can be represented using a graph, both objective can be modelled and solved as a balanced maximal independent set problem.

For example, any scheduling problem. Suppose a computational task is divided into various subtasks that can be run in parallel. Each subtask makes use of any one of the shared memory from a set of distinct independent shared memory. Therefore, the subtasks using the same shared memory for reading and writing should not be executed simultaneously else there would be conflict. So, how do we divide the subtasks that can be executed simultaneously so that there is maximum output at each time slot and each processor is doing comparably equal work at a particular time slot. Such situations can be composed using a graph and solved by decomposing the graph into balanced maximal independent sets.

The standard graph problem of generating maximal independent sets (MIS) focused on having minimum number of MIS generated [1]. But most of the practical algorithm will result in highly skewed maximal independent sets i.e. there will be considerable difference between the sizes of MIS generated. This causes a heavy imbalance in the work load of the processors.

In this paper, we have come up with a heuristic to address a variant of the problem of decomposing the graph into balanced maximal independent sets. Since, obtaining perfectly balanced maximal independent sets is a NP-hard problem; we address the problem, equitable [10] maximal independent sets, in which each maximal independent set has comparably equal size. For this, we will make use of the Monte Carlo's maximal independent set algorithm [8]. In an undirected graph  $G = (V, E)$ , an independent set (IS) is a collection  $I$  of vertices such that no two vertices in  $I$  are adjacent and an independent set  $I$  is a maximal independent set (MIS) if for  $v \in V$ , one of the following is true:

1.  $v \in I$
2.  $N(v) \cap I^c = \emptyset$  where  $N(v)$  denotes the neighbours of  $v$ . i.e. a vertex either belongs to the independent set or has at least one neighbour vertex that belongs to the independent set. As a result, every edge of the graph has at least one endpoint not in  $I$ . However, it is not true that every edge of the graph has at least one or even one endpoint in  $I$ . We

will use the parallelization technique for both generating and balancing the maximal independent sets obtained.

In the following sections, we will discuss about the previous work done in this field, the algorithm and analysis which we have proposed. We will also compare the results obtained in following sections.

## 2. BACKGROUND

Dividing a graph into several maximal independent sets is arranging of vertices of the graph into several subsets such that there are no pair of vertices into a subset who are sharing a common edge between them and no other vertex can be added into a subset without destroying the independent set property. Two vertices share the same edge if they are in neighbourhood. The division of graph into maximal independent sets is said to be equitable [10],[6] when the difference in the number of vertices between any two sets is at most one. This is an NP-hard problem since it can be easily reduced to the NP problem of dividing a set of numbers into half with sum in each half is differed by at most one [5]. Therefore, obtaining equitable maximal independent sets from a graph is costly to attain and most of the practical application does not demand ideal output. To enhance the performance, we compensate with the ideal output and define a closely related heuristic for balanced maximal independent sets as the difference in the size of the maximal independent sets can be at most a “small” number greater than 1. That is, the difference is bounded by a threshold number  $t$ . Bodleander and Fomin [3] proved that this problem can have a polynomial time solution.

Balanced maximal independent sets have found their importance in various areas. For example, parallel sparse matrix computations on irregular grids [9], register allocation problem [11], and load balanced partitioning for domain decomposition methods [12], various types of scheduling problems like class scheduling, process task scheduling and timetabling problems [2]. Various parallel programming algorithms require their task to be equally distributed to each processor. Balanced maximal independent sets prove to provide a great hand in enhancing the performance.

### 2.1 Previous Work

Lately a lot of effort has been put upon to address this problem. In this section, we will discuss the generating and balancing of maximal independent sets proposed by Hao Lu and Mahantesh [7], they have used parallel programming for balancing the maximal independent sets formed out of a graph. They used vertex centric approach for balancing the sizes of the bins.

Since dividing the graph into balanced maximal independent sets is an NP-hard problem but greedy scheme defined in Algorithm 1 [7] is found to be efficient in practical cases. This fundamental scheme is incorporated with various strategies to improve the performance and result.

### 2.2 Algorithms for Balanced Independent Sets

In this section, we will discuss about various algorithms which has been proposed for balanced independent sets. A number of strategies have been proposed to tackle the question which colour is to be used to colour the node (line 4 of Algorithm 1). Various approaches of two different types have been defined. In the first type, the goal is to obtain balanced independent sets in one step. We call such strategies

---

#### Algorithm 1: GreedyScheme

---

```

Procedure Begin
    GreedyScheme ( $G = (V, E)$ )
    for each vertex  $u$  in  $V$ , in some order do
        | Allocate a color to vertex  $u$  that is not allocated to
        | any of its neighbors
    End

```

---

as Ab initio strategies. In the second type, the goal of balanced independent sets is obtained by a following a two-step procedure. In the first step, the graph is decomposed into various independent sets and then is balanced as a part of the next step. We call such second type as guided strategies [7].

#### 2.2.1 Ab initio strategies

In this type, the strategies used for decomposing the graph into balanced independent sets do both generating and balancing together in one step. One such strategy is Greedy-LU (Least Used). In this strategy, the independent set to which the vertex will be assigned to in line 4 of Algorithm 1 is decided by choosing the independent set which has minimum size. Therefore, balancing is also taken care with the assignment. The pros of such strategy include that we don't have to do extra step for balancing. The generated independent sets obtained are roughly balanced. But the cons is that the number of independent sets formed may be quite high than the minimum number of independent sets required. Therefore, such sequential strategies do quite good when there is no limit on the number of independent sets or when the graph is sparse.

#### 2.2.2 Guided strategies

One such strategy is to colour the node with the least index available. This strategy is called as First Fit (FF) i.e. assigning the node to the first set it could place in. This analogy is used because it assure that the maximum different colours used are not more than  $\Delta + 1$  ( $\Delta$  = maximum degree of a node). But the major disadvantage of this strategy is that the sets obtained are highly skewed i.e. the difference between the size of the sets is very high. Various other strategies are also available, some of them are defined in the table 1.

In this guided strategy, initially the graph  $G = (V, E)$  is divided into various independent sets. Let the number of independent sets obtained be  $n$ . Then, a variable  $avg = |V| / n$  is defined which is the number of vertices that each independent set should contain for it to be called as balanced. The number of vertices in each independent set should be comparable to the number  $avg$ . The sets having size more than  $avg$  are termed as over-full bins and the sets having less size than  $avg$  are termed as under-full bins. A various algorithms, both parallel and sequential, are defined for transferring the vertex from over-full bins to under-full bins [4]. The assignment of colour can be any of the approach defined in table 1.

A parallel vertex centric approach is defined in the paper [7] for balancing the independent set. In this algorithm, the balancing step is done by randomly transferring the vertex from one over-full bins to each under-full bins one by one in order of increasing size. However, random transfer could cause a conflict, therefore, the transferred vertices are taken

**Table 1: List of strategies for balanced independent sets [7]**

Strategy	Category	Description
Greedy-LU (Least Used)	ab initio	Choose LU color among allowable colors and run algorithm 1.
Greedy-Random	ab initio	Choose a Random color among allowable colors and run algorithm 1.
Shuffling-unscheduled	guided	With First Fit strategy, run algorithm 1. Distinguish the over-full and under-full sets on the basis of obtained coloring . Shift the vertices from over-full to under-full sets keeping number of color classes same. Other specializations comprise Vertex-centric FF (VFF) and Color-centric LU (CLU).

back from under-full bins if there is a conflict. Then again, the algorithm repeats itself for achieving balanced IS.

---

**Algorithm 2:** Vertex Centric parallel scheme for balancing IS ( $G = (V, E)$ )

---

```

Procedure Begin
    Obtain the independent Sets
    Let  $V1$ , the set of vertices belong to over-full bins
    while  $V1 \neq \phi$  do
        for  $v$  in  $V1$  do
            do in parallel
            Let  $x$  as the smallest index of an under-full bin
            that is allowable
            if  $x$  exists then
                 $\text{color}[v] = x$ 
                Update size of bin  $x$ 
         $S = \phi$ 
        for  $v$  in  $V1$  do
            do in parallel
            for  $u$  in  $\text{adj}(v)$  do
                if  $(\text{color}[u] = \text{color}[v] \text{ and } u < v)$  then
                     $S \leftarrow S + v$ 
         $V1 \leftarrow S$ 
End

```

---

In above approaches, the generating IS step is sequential but in our approach, we try to parallelize both the generating and balancing steps. In balancing step, in above algorithm, since each IS from over-full bin chooses one under-full bin and rest of IS remain idle. In our algorithm, we try to engage all the possible under-full IS and over-full IS in each iteration to increase the performance. In the next section, we will discuss about our algorithm to address the problem.

### 3. PROPOSED HEURISTIC

In the previous section we have distinctly defined the problem statement i.e. balancing the maximal independent sets, generated out of a graph. So we can distinctly divide the problem statement into two sub-problems:

1. Generating the Maximal independent sets from the graph,
2. Balancing the generated maximal independent sets.

In this section we will explain the proposed heuristic to tackle the problem one by one.

#### 3.1 Generating MIS

Generating a Maximal Independent Set out of an undirected graph  $G = (V, E)$  is a common problem addressed

by Monte Carlo algorithm [8]. In this MIS algorithm (Algorithm 3), the input is the undirected graph  $G = (V, E)$  and the output is a maximal independent set. Monte Carlo algorithm can be run in parallel in hadoop environment using apache Giraph as pregel like system. Suppose  $G' = (V', E')$  be a subgraph of  $G$ . The subgraph  $G'$  contains all the possible vertices which can contribute to the maximal independent set,  $I$ . In each iteration, we find a maximal independent set  $I'$  out of the subgraph  $G'$  which will be added to the maximal independent set,  $I$ . The neighbourhood of the maximal independent set  $I'$  is removed from the subgraph  $G'$  because according to the definition of the subgraph of  $G'$ , only the vertices which has a possible chance of contributing to the maximal independent set  $I$  will be in  $G'$ . Since,  $I'$  has been added to the maximal independent set  $I$ , its neighbourhood then will not have a possible chance of coming to the maximal independent set  $I$  as otherwise it violate the condition of the maximal independent set. The  $N(I')$  represent the neighbourhood of the maximal independent set  $I'$ .

---

**Algorithm 3:** MIS ( $G = (V, E)$ )

---

```

Procedure Begin
     $I \leftarrow \phi$ 
     $G' = (V', E') \leftarrow G = (V, E)$ 
    while  $G' \neq \phi$  do
        do in parallel
        select a maximal independent set  $I' \in V'$  which is in
         $G'$ 
         $I \leftarrow I + I'$ 
         $X \leftarrow I' + N(I')$ 
         $G'' = (V', E')$  is induced subgraph on  $V' - X$ 
End

```

---

For finding the maximal independent set  $I'$  in line 6, we propose the following heuristic:

1. Take all the vertices in the graph  $G'$  into the maximal independent set  $I'$ .
2. For all the vertices present in the set  $I'$ , validate the independent set to make it as an independent set by removing the adjacent vertices.

**Validation step:** Validation is done by removing any one of the vertices which are present in the set and are sharing same edge. The decision for deciding which vertex is to be removed lies on the degree of the vertex i.e. vertex with lowest degree is removed. If the degree of both the vertex is equal then we choose the vertex having vertex id greater than the other. Vertex Ids are randomly allocated to each

vertex and is fixed throughout the computation.

Once the while loop is terminated, we will have the set MIS as our output which is the required maximal independent set. Now, to divide the entire graph into sets of maximal independent sets (Algorithm 4), we will repeat the above process again and again as follows. Once we obtain the set MIS, we will update the original graph G with  $G - MIS$  and repeat the above process to find another maximal independent set till graph G becomes empty. Once the graph G became empty, we will have the required output which we need to balance.

---

**Algorithm 4:** MISGenerator

---

```

Procedure Begin
while  $G \neq \emptyset$  do
     $G' = (V', E') \leftarrow G = (V, E)$ 
    MIS = MIS( $G'$ )
    Give MIS a unique ID
    Update  $G \leftarrow (G - MIS)$ 
End
```

---

### 3.2 Balancing Maximal Independent Sets

In this subsection, we describe the heuristics to obtain balanced maximal independent sets of a given graph. The first step is obviously to get the graph divided into various maximal independent sets. To obtain the initial maximal independent sets we consider the effort used in the previous step (Algorithm 4). The output maximal independent sets from the previous step act as input to this step for balancing the obtained maximal independent sets.

The heuristic for balancing maximal independent sets can be defined as follows: Given a graph  $G = (V, E)$  where V represents the vertex set of the graph and E represents the edge set of the graph, let the number of maximal independent sets found be NumOfMIS. We define a number 'avg' which is the total number of vertices that each maximal independent set should have to be called as balanced maximal independent sets. Therefore, avg can be defined as,  $avg = |V| / \text{NumOfMIS}$ . Now we compare the size of each maximal independent set with this avg and partition the maximal independent sets into two sets named as MinSet and MaxSet such that MinSet contains the maximal independent sets whose size are less than the number avg and MaxSet contains the maximal independent sets whose size are more than the number avg.

Since balancing maximal independent set is NP-hard problem, we will address the balancing maximal independent set as follows: How should we transfer a set of vertices from maximal independent sets in MaxSet to the maximal independent sets in MinSet such that after rearranging, all the maximal independent sets have the number of vertices roughly equal to 'avg'.

We have proposed the following heuristic for the problem statement described above. Consider each maximal independent set in sets MinSet and MaxSet as individual nodes. Now build a graph  $G' = (V', E')$ . We define the graph  $G'$  as follows:  $V'$  are the nodes representing the maximal independent sets from sets MinSet and MaxSet and the  $E'$  is edge set such that each node from MinSet has an undirected edge to the each node in MaxSet. The weight,  $W(i,j)$  of the each edge  $E'$  is the number of vertices that the node j from

MaxSet can donate to the node i from MinSet. The numbers of nodes which will be donated are defined on several factors such as:

1. The size of the maximal independent set which is donating does not fall less than avg.
2. The size of the maximal independent set which is receiving the vertices must not rise more than the number avg.
3. The definition of the maximal independent set should be satisfied after the shifting of the vertices.

Therefore, the weight of an edge between two maximal independent set i in MinSet and j in MaxSet will be calculated as follows:

$$\begin{aligned} & \text{Number of possible candidates for shifting} \\ &= |\text{number of vertices in } j| - |N(i,j)| \end{aligned}$$

where  $|\text{number of vertices in } j|$  are the total number of vertices in maximal independent set  $j$  and  $|N(i,j)|$  is the number of vertices in  $j$  such that these vertices are in neighbourhood of maximal independent set  $i$ . This will take care of the factor 3 described above. To satisfy the factor 1 and 2, we must take into account the number of vertices in each MIS  $i$  and  $j$ .

Therefore, the weight of the edge will be:

$$\begin{aligned} W(i,j) &= \min \{ \text{number of vertices required by set } i, \\ & |\text{number of vertices in } j| - avg - \\ & \max \{ 0, avg - N(i, j) \} \} \end{aligned}$$

This will be the weight of edge between any two nodes  $i, j$  of graph  $G'$ . Once we have defined our hypothetical graph  $G'$ , now we will address the main problem which is shifting the vertices from MaxSet to MinSet. At each iteration, one MIS from MinSet is paired with one MIS from MaxSet. Then, the shifting of vertices takes place. The number of vertices shifted is the weight of the edge between each pair. This step is done in parallel. Then, the edge weights between each maximal independent set in MinSet and MaxSet are updated since the number of vertices in each maximal independent set has been changed after shifting.

The pairing of one MIS from MinSet to another MIS from MaxSet is done via Bin Packing Algorithm [4] i.e. according to the weight of the edges between each independent sets of graph  $G'$ , the MIS which can donate maximum number of vertices is paired with the MIS with minimum number of vertices. This is done until all the maximal independent sets from MinSet is paired with MaxSet. Finally, the shifting of vertices takes place. These iterations repeat over and over till we reach a saturation point i.e. no more vertices can be transferred (Algorithm 5).

## 4. ANALYSIS

In this section, we will talk about the time complexity of the proposed algorithm. As per the [8], EREW P-RAM has  $O(m)$  processors and time taken by each execution is  $O(\log n)$ . In [8], it has been proven that the total number of executions of the while loop before the algorithm terminates is  $O(\log n)$ . Hence, the implementation of the algorithm on an EREW P-RAM takes  $O(m)$  processors and the total execution time of the algorithm is  $O((\log n) * (\log n))$ .

The balancing of maximal independent set takes  $O(k^k)$  time for allocating the weights to the edges of the graph  $G'$

---

**Algorithm 5:** Balncing MIS

---

Procedure **Begin**  
Get maximal independent sets by calling MISGenerate function  
Let U be the bins from MinSet and O be the bins from MaxSet  
**while**  $MinSet \neq \phi$  **AND**  $MaxSet \neq \phi$  **do**  
    **for**  $u \in U$  **do**  
        **for**  $v \in O$  **do**  
            Define an edge  $u-v$  with weight of edge,  
             $W(u, v) = \min(\text{avg} -$   
             $\text{NumOfVertices}(u), \text{NumOfVertices}(v) -$   
             $\text{avg} - \max(0, N(u, v) - \text{avg}))$   
    Sort bins in Minset in increasing order of  
    NumOfVertices  
    **for**  $u \in U$  **do**  
        Select  $v \in V$  such that  $W(u, w) < W(u, v)$  where  
         $w \in V$  and  $v, w$  are not selected by any  $u \in U$   
        till now  
         $u \leftarrow u + W(u, v)$   
         $v \leftarrow v - W(u, v)$   
**End**

---

as described in subsection 3.2, where  $k$  is the total number of maximal independent sets formed. The total number of maximal independent sets formed is based on the density of the graph taken as input. In worst case, we get total number of independent set equal to the number of vertices in the input graph, this happens when the graph is a clique. In best case, the number of independent sets formed are  $O(1)$ , when the number of edges in the input graph is very less than the number of the vertices. But according to the reference [8], on average  $O(\log n)$  number of maximal independent sets are formed from a sparse graph. The step of detecting the neighbourhood and moving of vertices takes a maximum of 2 to 3 substep, therefore it takes  $O(1)$  complexity. The number of times the steps of defining, marking and shifting of vertices takes place till the computation reaches a saturation point (i.e. no shifting of vertices is possible which will lead to balance the MIS) is  $O(\log n)$ . The proof for this is as follows: There are  $O(\log n)$  number of MIS formed. In each iteration, a MIS from MinSet is paired with MIS from MaxSet and the number of vertices between this pair will lead to the following possibilities: 1. MIS from MinSet will reach to avg number of vertices, 2. MIS from MaxSet will reach to avg number of vertices, 3. Both MIS from MinSet and MaxSet will transfer maximum vertices but still both of them have room to receive vertices and donate vertices respectively. Since after shifting the vertices, the result is one of the three possibilities stated above. We claim that, these two sets will never be paired again in future for balancing. Since, if the result is the 1st or 2nd possibility, then at least one of the MIS out of them will be refrained from receiving or donating the vertices. If the result is the 3rd possibility, then also they cannot be paired again since if they are pairing in future then they can shift vertices, but if they are able to shift vertices in future then they must have shifted when the first time they paired. Therefore, every MIS in MinSet can pair every MIS in MaxSet at most one. Since, the number of maximal MIS formed are  $O(\log n)$ , therefore the number of times the steps of defining the

graph  $G'$ , marking and shifting of vertices will take place at most  $O((\log n) * (\log n))$  times. Therefore the total complexity for balancing the maximal independent set is  $O((\log n) * (\log n) * (\log n) * (\log n))$ .

## 5. EXPERIMENTAL SETUP AND RESULT

We have used hadoop for implementing the proposed algorithm. In hadoop, a cluster of 2 nodes was established. The RAM given to the nodes was 2 GB and the hard disk size was 20 GB. The operating system used for experimentation was ubuntu genome.

The dataset used for the experimentation was taken from the SNAP (Stanford Network Analysis Project). A brief detail of the dataset is given below.

**Table 2: Generating Maximal Independent Sets**

Generating Maximal Independent Sets	
Number of Maximal Independent Sets formed	9
Number of supersteps	78
Aggregate edges	1,851,744
Aggregate sent messages	72,300,892
Aggregate vertices	334,863

## 5.1 Results

The proposed algorithm for generating and balancing the maximal independent set was run on the dataset described above. In the section below, we will compare the results of our proposed approach with other approaches. Table 2 and Table 5 show the experimental result. The graph was divided into a total of 9 maximal independent sets. Once we have generated the maximal independent set, the result of this was given as input for balancing the sets. The result we obtain is briefly described in the Table 2 and Table 5. At each iteration, the maximal independent sets from MinSet were paired with maximal independent set from MaxSet. The pairing result is given in the Table 2.

### 5.1.1 Generating MIS

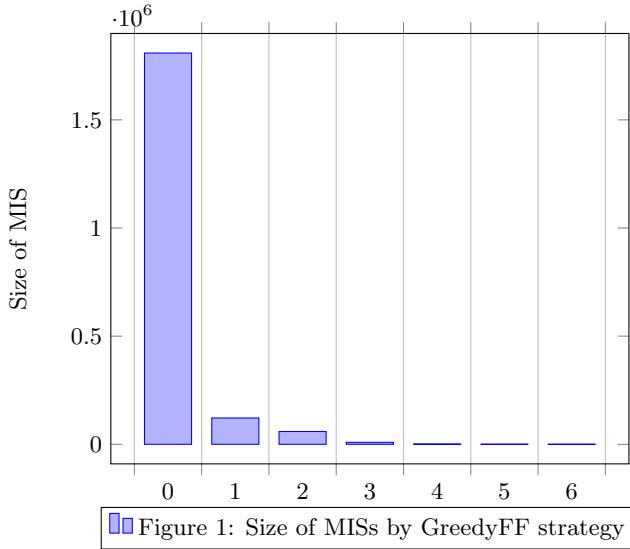
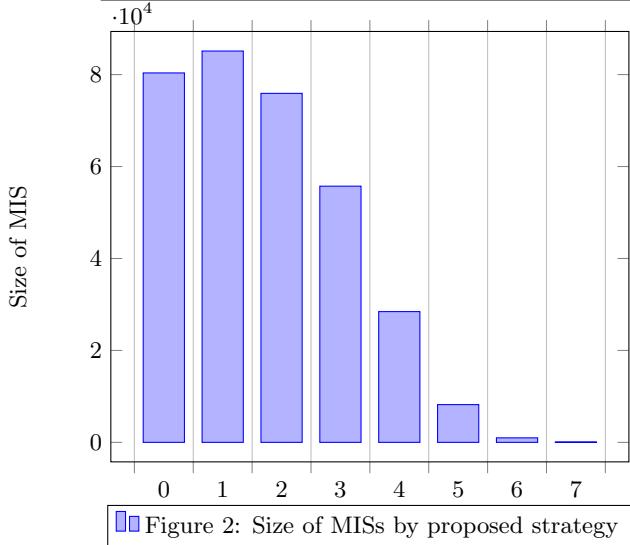
Since our approach is a guided strategy type i.e. first the MIS are generated and then these sets are balanced. According to the heuristic discussed in section 3, maximal independent sets are generated from the graph. The total numbers of maximal independent sets obtained are 9. A detailed comparison of the size of each MIS is shown in Table 3. GreedyFF (First Fit) and GreedyLU (Least Used) approaches are also run against the same dataset. A fine comparison of the output obtained from these approaches can be seen using the Figure 1 and Figure 2. The total numbers of MIS obtained by applying GreedyFF approach is minimum i.e. 8 but the size of the MIS obtained are highly skewed. On the contrary, the size of MIS obtained from GreedyLU approach is balanced but the total number of MIS obtained is quite high. But the size of the MIS obtained from our heuristic are fairly comparable. Also, the total number of MIS formed is also comparable to GreedyFF approach i.e. 9.

**Table 3: Balancing Maximal Independent Sets**

Iterations	Maximal Independent Set over each Iteration								
	MIS-0	MIS-1	MIS-2	MIS-3	MIS-4	MIS-5	MIS-6	MIS-7	MIS-8
0	80364	85129	75922	55732	28443	8210	979	74	10
1	43167	48901	38789	37207	28443	26735	37207	37207	37207
2	37207	38429	38789	37207	34403	37207	37207	37207	37207
3	37207	38429	37207	37207	35985	37207	37207	37207	37207
4	37207	37207	37207	37207	37207	37207	37207	37207	37207

**Table 4: Balancing Accuracy**

Input Graph	Perc. Difference in size before balancing	Perc. Difference in size after balancing	No. of MIS (GreedyFF)	No. of MIS (Proposed Strategy)
com-Amazon	88.5	0.00	8	9
Wiki-Vote	70.3	0.00	4	4
Email-EuAll	98.21	0.00	3	3
Random Dense network	57.76	0.04	9	10


**Figure 1: Size of MISs by GreedyFF strategy**

**Figure 2: Size of MISs by proposed strategy**
**Table 5: Comparison of Generated Maximal Independent Sets by different approaches**

Generating Maximal Independent Sets		
Strategy	Number of MIS	Balance Status
Maximal IS	9	Fairly balanced
GreedyFF	8	Highly unbalanced
GreedyLU	39	Balanced

### 5.1.2 Balancing Maximal Independent Sets

The next task was to balance the maximal independent sets obtained. The MIS sets obtained were balanced by following the algorithm for balancing MIS as discussed in section 3. The MaxSet contains the MIS 0, 1, 2, 3, 4 and MinSet contains the MIS 5, 6, 7, 8. A total of 4 iteration were executed for balancing. Table 6 shows the pairing done from MIS of MinSet to MIS of MaxSet in each iteration. Table 3 shows the size detailed summary i.e. size of each MIS after each iteration. Table 4 shows balancing accuracy for different input graphs.

## 6. CONCLUSIONS

In this paper, we have provided a heuristic for obtaining balanced maximal independent sets from the graph. We also compared the time complexity and output of the algorithm proposed against a set of techniques which has been proposed earlier. Specifically, we have presented a parallelize algorithm with optimized number of balanced maximal independent sets. Maximal independent sets play an important role in identifying independent tasks in a number of parallel computing applications. We expect that this paper will give a valuable reference to the applications developers who are looking for various approaches to improve the parallel performance.

**Table 6: Balancing MIS**

Pairing b/w <i>MinSet</i> and <i>MaxSet</i>		
Iterations	MaxSet	MinSet
0	0	8
	2	7
	1	6
	3	5
1	0	4
	1	5
2	2	4
3	1	4

## 7. REFERENCES

- [1] N. Alon, L. Babai, and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of algorithms*, 7(4):567–583, 1986.
- [2] J. Blažewicz, K. H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz. *Scheduling computer and manufacturing processes*. Springer Science & Business Media, 2013.
- [3] H. L. Bodlaender and F. V. Fomin. Equitable colorings of bounded treewidth graphs. *Theoretical Computer Science*, 349(1):22–30, 2005.
- [4] E. G. Coffman Jr, M. R. Garey, and D. S. Johnson. Approximation algorithms for bin packing: a survey. In *Approximation algorithms for NP-hard problems*, pages 46–93. PWS Publishing Co., 1996.
- [5] H. Furmanczyk and M. Kubale. Equitable coloring of graphs. *Contemporary mathematics*, 352:35–54, 2004.
- [6] A. Hajnal and E. Szemerédi. Proof of a conjecture of erdos. *Combinatorial theory and its applications*, 2:601–623, 1970.
- [7] H. Lu, M. Halappanavar, D. Chavarría-Miranda, A. Gebremedhin, and A. Kalyanaraman. Balanced coloring for parallel computing applications. In *Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International*, pages 7–16. IEEE, 2015.
- [8] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM journal on computing*, 15(4):1036–1053, 1986.
- [9] R. G. Melhem and K. Ramarao. Multicolor reordering of sparse matrices resulting from irregular grids. *ACM Transactions on Mathematical Software (TOMS)*, 14(2):117–138, 1988.
- [10] W. Meyer. Equitable coloring. *The American Mathematical Monthly*, 80(8):920–922, 1973.
- [11] F. M. Q. Pereira. A survey on register allocation. Technical report, Technical report, University of California, United States of America, 2008.
- [12] B. Smith, P. Bjorstad, and W. Gropp. *Domain decomposition: parallel multilevel methods for elliptic partial differential equations*. Cambridge university press, 2004.

# Reinforcing Immutability of Permissioned Blockchains with Keyless Signatures' Infrastructure

Nitesh Emmadi and Harika Narumanchi

TCS Innovation Labs

Hyderabad, India

{nitesh.e,harika.n}@atc.tcs.com

## ABSTRACT

With the emergence of Bitcoin, businesses are focusing on leveraging Bitcoin's blockchain technology to non-cryptocurrency based applications to improve efficiency of the operations. These business applications operate in environments where participants have verified identities; these are called permissioned environments. Blockchain is an immutable and append only distributed ledger that can be utilized for record keeping applications. We observe that when blockchain technology is adapted from permissionless environments to permissioned environments the immutability of blockchain becomes questionable as the end-users may not monitor or store a copy of the blockchain. We propose the use of Keyless Signatures' Infrastructure as an additional mechanism to ensure irreversible and irrefutable proof of block confirmations which acts as a global proof thus preserving long term immutability of blockchain in permissioned environments.

## Keywords

Bitcoin; Blockchain; Permissioned Environments; Permissionless Environments; Keyless Signatures' Infrastructure; Merkle Tree; Hashchain

## 1. Introduction

In 2008, Satoshi Nakamoto proposed a type of peer-to-peer decentralized digital currency termed as Bitcoin [1] based on a mathematical proof, created and maintained electronically. Bitcoins are mined by a group of individuals called *miners* through a process called *mining*. The difficulty in mining is introduced by the *proof of work* [2], an operation that is expensive and time consuming to execute but easy to verify. Miners validate the pending transactions in the Bitcoin network - submitted during a time period - into a *block* and confirm those transactions into a public ledger called the *blockchain*. Blockchain is an immutable and ap-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ICDCN '17, January 04-07, 2017, Hyderabad, India

© 2017 ACM. ISBN 978-1-4503-4839-3/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3007748.3018280>

pend only distributed ledger predominantly used for record keeping. Most recently, the progress in blockchain technology for use in applications other than cryptocurrencies has been promising due to advantages such as anonymity, decentralization, speed, non-repudiability, transparency and difficulty to forge giving them a unique edge over alternative solutions. For example, blockchain can be utilized to create and transfer digital assets in an efficient manner. It can also be used to build infrastructure for trade finance.

Several computational platforms were proposed on top of blockchain such as Enigma [3] and HAWK [4] that operate in open environments where the identity of the participants is anonymous. Enigma is a decentralized computational platform that ensures privacy. Its computational model is based on secure multi-party computation guaranteed by verifiable secret sharing schemes. HAWK is a decentralized model for privacy preserving smart contracts that ensures on-chain privacy and contractual security.

Most recently, there has been ongoing research on designing blockchain platforms that operate in permissioned environments where the identity of the involved parties is known. The transaction validators in open environments are anonymous outsider entities whereas permissioned environments have validators whose identities are required to be verified before joining the network. Organizations are developing their own blockchains solutions to support a wide range of applications. For example, Ethereum [5] is one of the decentralized smart contract platforms that can execute peer-to-peer contracts using cryptocurrency called *ether*. The Linux Foundation announced the creation of Hyperledger project [6] which is a cross-industry collaborative effort with more than hundred contributors to create an open standard for blockchain. Chain.com [7] supports asset transfer that has automated rules for issuance and transfer with role-based permissions for accessing the network. Eris blockchain platform [8] focuses on building financial applications leveraging smart contracts.

In this paper we observe that adapting blockchain to permissioned environments has an impact on the immutability property of blockchain. We propose the use of Keyless Signatures' Infrastructure (KSI) [9] signatures to provide irrefutable proof of block confirmations thus strengthening the immutability of blockchain. KSI provides a hash-tree based time-stamping service that provides a proof that the data existed at a particular time.

### 1.1 Our Contribution

We observe that adapting blockchain to permissioned en-

vironments affects the immutability of blockchain with higher probability compared to permissionless environments. The validating parties can collude at some point to change the ledger (include or exclude transactions) and reconstruct it. The customers may not store the blockchain due to large number of transactions and memory constraints, or due to implicit trust in the validating parties. As the ledger is maintained by the controlled entities, we cannot ensure the immutability of the ledger. We propose the use of KSI signatures as an additional mechanism to ensure immutability wherein the security of the system does not depend on long-term secrecy of private keys as there are no keys involved in signature generation. The KSI signature acts as a receipt of the block hash, which includes user's transactions, and can be used to verify the integrity of the blockchain. This mechanism provides a way of verifying the integrity of the blockchain ledger even to the users who do not want to monitor the blockchain continuously. The past signatures in the KSI hashchain cannot be modified or generated. As KSI hash tree is public, the difficulty of forging a signature, is much higher and can be easily detected. KSI maintains a global proof that the authorities of the blockchain has no control over, thus ensuring that the ledger has not been tampered with.

## 1.2 Preliminaries

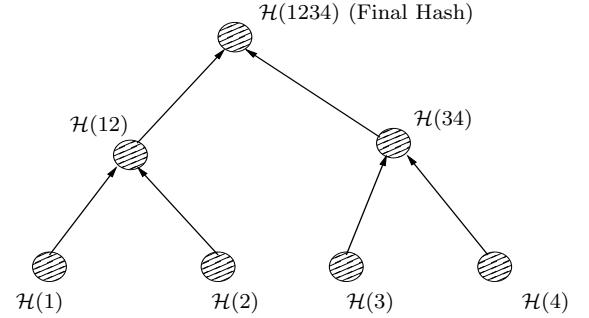
In this paper public key is represented by  $pk$  and private key is represented by  $sk$ . The  $n^{th}$  block in a blockchain is represented by  $b_n$ . Hash of a statement  $x$  is represented by  $\mathcal{H}(x)$ . A hash of a block  $b$  verified by both KSI signature and digital signature is represented by  $\mathcal{H}_v(b)$ . The  $n^{th}$  transaction is represented by  $t_n$ . Proof of work is denoted by  $P_w$ .

The paper is organized as follows: Section II describes the blockchain in Bitcoin and how immutability is achieved in Bitcoin. Section III covers the challenges involved in adapting blockchain from permissionless to permissioned environments. Section IV illustrates KSI and how it can be used to strengthen the immutability of blockchain in permissioned environments.

## 2. Blockchain in Bitcoin

A hashchain is formed by recursive application of a hash function to a stream of data. In hashchain, previously computed hash is included in the current hash. The interesting property in hashchains is once the final hash is computed and confirmed, it is infeasible to alter the previous data without modifying the final hash. The confirmed final hash acts as a validator for the entire chain. When a hashchain becomes very long, the verification process takes lot of time. To overcome this disadvantage, we group some hashes before using them in the main tree to expedite the verification process, termed as the Merkle tree [10]. A Merkle tree is an optimized hashchain and is constructed by hashing pairs of leaves until a single hash remains, the Merkle root. Merkle tree is an efficient way to verify large data structures. Merkle trees are used in different types of softwares like file sharing protocols such as BitTorrent, distributed version control systems such as Git and blockchain in Bitcoins.

Blockchain is a chain of a continuously growing list of transaction records. It is essentially tamper-proof and pro-



**Figure 1: Merkle Tree**

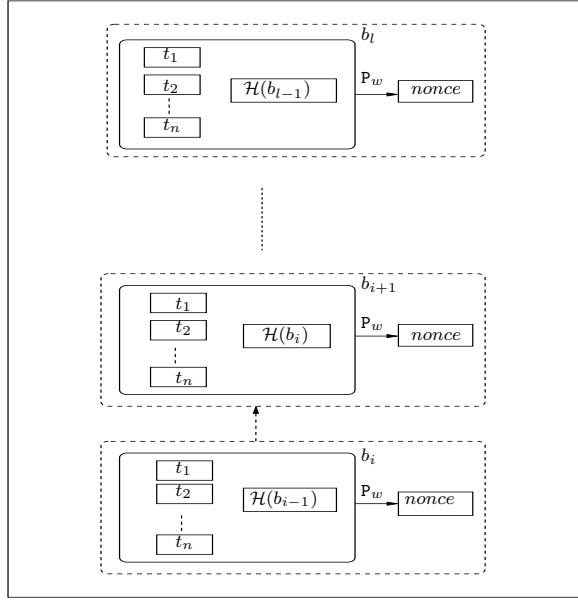
vides integrity and transparency along with efficient availability [1]. The content of the blockchain can be accessed by anyone and the transactions of the blockchain can be verified anytime. The blockchain has to be updated with the recent changes since the last update. Each block is computationally infeasible to modify once it has been in the chain for sometime. To modify a block, every block after it has to be regenerated and the proof-of-work computation has to be redone. We refer to the computed hash, using the current hash and previous hash, as the *Final hash* (Figure 1). When a final hash is processed by the network and is highly unlikely to be reversed, we refer to it as the *confirmed hash*. The risk of reversing a confirmed transaction exponentially decreases with each subsequent confirmations.

Bitcoin network uses proof-of-work to ensure consensus among the Bitcoin miners. Proof-of-work is used in trustless environments where there is no central authority and mutual trust. The first proof of work system was invented in 1997 called *Hashcash* [2]. It was used to limit spam emails and denial of service attacks. The proof of work function used as a mining algorithm in Bitcoin, uses essentially the same idea of Hashcash. The authenticity of proof of work is based on computational difficulty and honest majority. This protocol involves a challenge, say  $c$  and the prover of the protocol comes up with a response, say  $r$  corresponding to the given challenge  $c$  that satisfies a definite mathematical property. The challenge  $c$  should be such that the prover takes significant amounts of CPU time or resources to compute the response, but takes a lot less time to verify. The proof-of-work mechanism is highly inefficient both in terms of energy consumption as well as transaction throughput.

Immutability in Bitcoin is due to the following reasons:

- Proof-of-work requires expensive and time consuming operation satisfying a certain challenge that requires enormous amount of computational power but is easy to verify. The adversary who tries to overwrite the response with a fake value would spend enormous amounts of CPU time or resources to successfully overwrite the legitimate value.
- The copies of the blockchain are distributed and maintained by several independent parties. Therefore, it is not feasible to modify the blockchain without the knowledge of all the parties.
- Blocks are appended periodically after every ten minutes. Therefore, to modify a particular block all blocks after it should be regenerated.

Figure 2 represents the working of blockchain in Bitcoin environment. Miners run the proof of work for the  $k^{th}$  block represented by  $P_w(b_k)$  to find the nonce that satisfies the given condition (decided by the Bitcoin network).



**Figure 2: Blockchain with proof of work**

### 3. Adapting Blockchain to Permissioned Environments

Having seen how the immutability is achieved in the permissionless environment such as Bitcoin [Section 2], this section describes how the permissioned environments differ and how the immutability is affected.

While adapting the blockchain to permissioned environments, difficulty in the consensus protocols in these environments can be, and often need to be, relaxed. There are challenges involved in adapting the consensus protocols used in permissionless environments to the permissioned environments. For example, the proof-of-work is not suitable in controlled environments due to its high energy consumption and low transaction throughput. Also, there is sufficient time for the attacker to reconstruct at-least the recent blocks in the blockchain during public holidays or weekends or auditing periods where new transactions are prohibited. In the case of Bitcoins this problem does not arise because even if a block has no transactions, miners mine a block every 10 minutes to collect rewards from it. Therefore, it is more difficult for the attacker to reconstruct the blocks as more blocks are periodically appended to the blockchain. However, in permissioned environments, as the frequency of the transactions is irregular, proof of work is not a suitable choice for consensus. It might seem a simpler solution to consider use of digital signatures to sign the verified transactions to reach consensus as the validating parties have verified identities.

However, as the copies of the blockchain are maintained by controlling parties, they can collude to modify the blockchain and reconstruct it. As the end-users may not monitor or

store a copy of the blockchain, we cannot ensure that the blockchain is not tampered with.

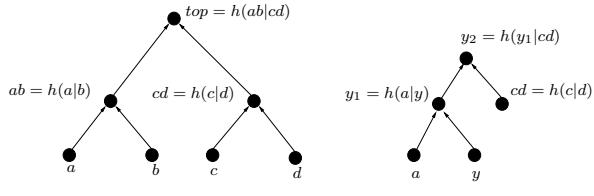
### 4. Blockchain and Keyless Signatures' Infrastructure

Considering the challenges in adapting blockchain from permissionless environments to permissioned environments, we need a mechanism that the controlling party does not have control over to ensure the immutability of the ledger. To facilitate this we use a global time-stamping mechanism to provide a global proof that the data exists at a point of time and has not been tampered with. We use KSI signature to generate the global proof. The hash-tree in KSI hashchain grows linearly with time. This property of the KSI hashchain makes it infeasible to modify or generate the predicated signatures facilitating its use in permissioned environments.

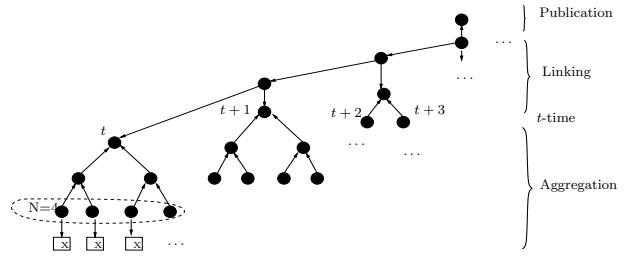
#### 4.1 Keyless Signatures' Infrastructure

KSI [9] provides scalable signature based authentication. KSI provides strong data integrity, long term immutability and tamper evident protection for digital assets with proof of time, identity and authenticity guarantee that the data has not been tampered with since it was signed. The integrity of the signatures is protected using one-way collision resistant hash functions and do not involve any secret keys. KSI hashchain is designed to provide scalable blockchains that scale linearly with time and independent of the number of transactions, unlike the traditional blockchains that scale linearly with the number of transactions. Hash-tree timestamping technique uses one-way collision resistant hash functions to convert a list of hashes into a fixed length hash associated with time. The end-user sends a hash to the service provider and receives a signature token. A signature token, which is essentially a path to the Merkle root, is the proof that the data existed at a specific point of time and the request was received through a particular access point. KSI aggregates all the received requests into a hash-tree and top hash values are retained for each second. These top hash values are linked together in a globally unique hash-tree called the hash calendar. In Figure 3, to verify a signature token  $y$  in the place of  $b$ , we concatenate  $y$  with  $a$  that is retained as a part of the signature token and compute hash value  $y_1 = h(y|b)$ .  $y_1$  is used as input for the next hash computation step until we reach the top hash  $y_2$ . If  $top = y_2$ , then it is risk-less to believe that  $y$  was in the original hash-tree. The size of the signature token depends on the number of aggregation levels. As the number of client requests increase, the number of aggregation levels increase logarithmically. Consequently, the number of hashes to be stored to construct the verification path to the root hash increases.

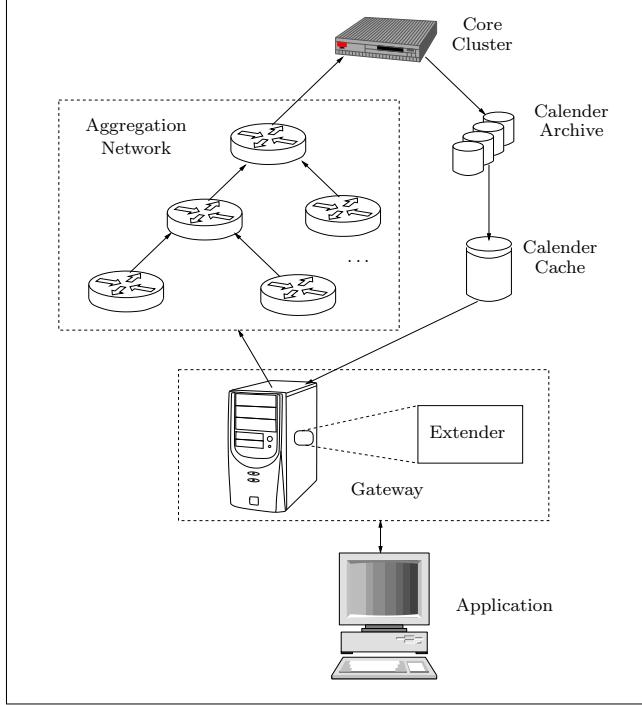
KSI works in three stages namely, hashing, aggregation and publication (Figure 5). The statements to be signed by KSI are hashed and the hash values represent the transactions. The user sends the hash of a statement through the KSI gateway to obtain signature in response to the hash. A global per-round hash is created using the aggregators to represent all the transactions signed during a round. The top values of the global per-round hashes are linked together into a perpetual hash-tree and published in a hard-to-modify and widely witnessed media such as newspapers.



**Figure 3: Building a hash-tree and verification of  $y$  in the position of  $b$**



**Figure 5: Hash-tree**



**Figure 4: KSI System Architecture**

The infrastructure design of KSI (Figure 4) consists of four major components namely application, gateway, aggregators and the core cluster. The application is used to perform the hashing step that forms the signing request. The signing request is sent to the gateway that provides services to the end-user. The gateway accepts the signing requests in application specific formats and forwards them to the assigned aggregators. The aggregators working in rounds of equal duration gather the incoming requests to build a hash tree and pass the top hash values to their upstream aggregators. The requests received during a particular round are aggregated into the same hashtree. An aggregator sends responses to all its child aggregators along with the hash path of its own tree after receiving a response from its upstream aggregator. A core cluster is on the top of the aggregation network and is responsible for achieving consensus about the top hash value in each round, storing it in calendar database and returning it to the aggregation network. There is an extender in the gateway that offers signature token verification services to the users. The values in the calendar database are distributed using calendar cache to the extender services. Refer to [9] for the detailed description of KSI architecture.

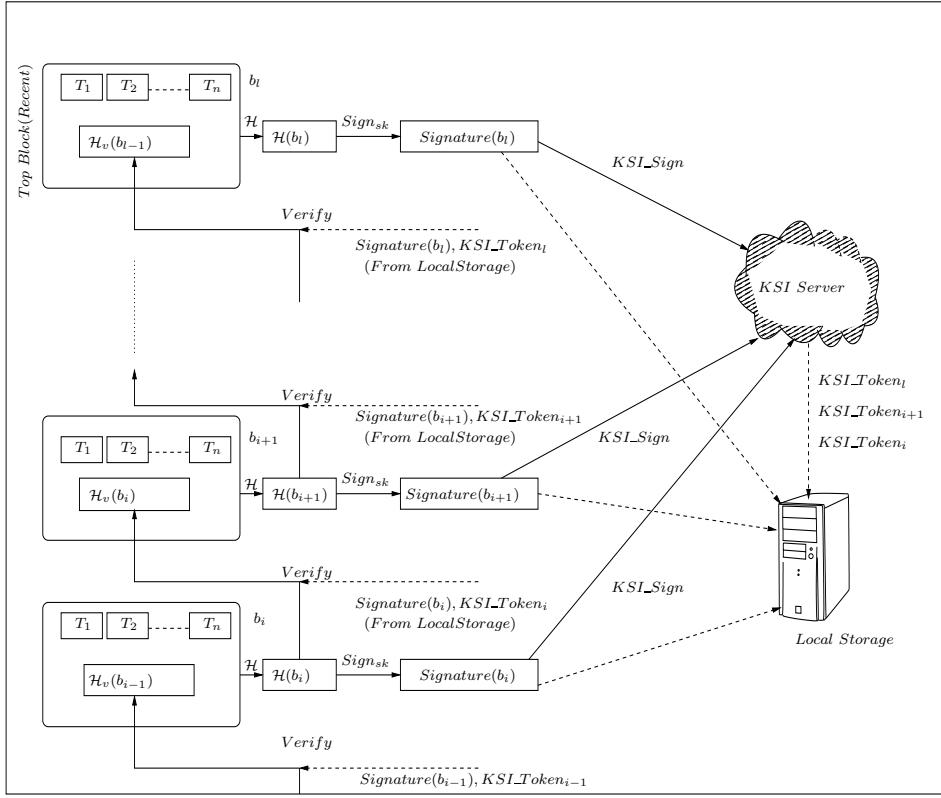
We conducted the test of KSI for blockchain time-stamping using the open-source library available on Github in Java. The KSI library and test access are available on request. When a block is signed using KSI, we obtain a signature token [Refer Appendix] in return which can be used to verify the block. The size of signature token file in this test example is 1757 bytes.

#### 4.2 Blockchain based on KSI

In the blockchain based on KSI hashchain (Figure 6), the hash of data is signed and sent to the KSI server as input. The KSI server signs this input and returns a signature token corresponding to the particular input, an irrefutable proof that the block existed. Note that the KSI server takes only the hashes of the data as input. Therefore, even if the server is compromised it is impossible to know what the data is. As the hashes are stored on a global hash calendar any modifications are globally visible, especially to other users of the KSI network. This signature token along with the hashes pertaining to that block can be given as a receipt to the user that the block existed which acts as a global proof for his transaction. Using the hashes pertaining to that block and the signature token, user can prove that the transaction was in a particular block on the blockchain at a point of time. It is the user's responsibility to store the signature token at the users' side. The end-user has global proof that the block existed at that point of time in the chain using the signature token of that particular block. The end-user can use this KSI signature token to verify the integrity of the blockchain without the need for monitoring or maintaining the blockchain. The main advantage of using KSI is that the security of the system does not depend on the long-term secrecy of the private keys. As KSI server takes only hash of data as the input, the privacy of the underlying data is preserved. It is impossible to compromise the Gaurdtime's global server infrastructure, without being detected. Therefore, it is infeasible to forge a KSI signature with past or future time-stamps.

## 5. Conclusion

Leveraging blockchain technology in enterprise applications is an active research area. Organizations are building their own blockchain infrastructure to improve their efficiency in performing operations. We addressed the issue of lack of strong immutability in permissioned environments due to the centralized authority that has the ability to modify the blockchain at some point of time. We proposed use of KSI signature to ensure irrefutable and irreversible proof



**Figure 6: Blockchain based on KSI hashchain**

of block confirmations to mitigate the problems in adapting blockchains to permissioned environments. The KSI signature ensures integrity of the blockchain even when the users do not monitor the blockchain.

## 6. Acknowledgment

We thank the team at TCS Innovation Labs, Hyderabad for extended discussions on this subject. We thank anonymous reviewers for their valuable suggestions in improving the paper.

## 7. References

- [1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.
- [2] Adam Back. Hashcash- a denial of service counter-measure. 1997. <http://www.cryptospace.org/~adam/hashcash/>.
- [3] Guy Zyskind, Oz Nathan, and Alex Pentland. Enigma: Decentralized computation platform with guaranteed privacy. *CoRR*, abs/1506.03471, 2015.
- [4] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. *University of Maryland and Cornell University*, 2015.
- [5] Ethereum Project. A next-generation smart contract and decentralized application platform, 2014.
- [6] Hyperledger. Hyperledger. 2015. <https://www.hyperledger.org/> (Accessed: 17-10-2016).
- [7] Chain. Chain. 2014. <https://chain.com/> (Accessed: 17-10-2016).
- [8] Eris Industries. Eris blockchain platform. 2014. <https://monax.io/platform/> (Accessed: 21-10-2016).
- [9] Ahto Buldas, Andres Kroonmaa, and Risto Laanoja. *Secure IT Systems: 18th Nordic Conference, NordSec 2013*, chapter Keyless Signatures' Infrastructure: How to Build Global Distributed Hash-Trees, pages 313–320. 2013.
- [10] Ralph Charles Merkle. Secrecy, authentication, and public key systems. 1979. AAI8001972.

## APPENDIX

The structure of KSI signature token from our test results can be seen below:

Attributes of the Signature Token:

Input Hash:SHA-256: [E3B0C44298FC1C149AFBF4C8996FB92427AE41E4649B934CA495991B7852B855]  
Aggregation HashChain Length:5

Aggregation Output Hashes:

Hash of Aggregation Chain[0]SHA-256: [3C80351B010E63EB497C6084767AD3B54E249B822871160D18B9222FOCC61145]  
Aggregation Chain Reference:com.guardtime.ksi.unisignature.inmemory.InMemoryAggregationHashChain@b99d62d9  
Number of ChainLinks in Aggregation Chain[0]:1  
Chain Link[0] Reference:com.guardtime.ksi.unisignature.inmemory.LeftAggregationChainLink@e307b26c

Hash of Aggregation Chain[1]SHA-256: [ADOFBE76FAFEB063A2CD7B3C0AB88106AF8984C53138CBDFFB3F29AB67873AA]  
Aggregation Chain Reference:com.guardtime.ksi.unisignature.inmemory.InMemoryAggregationHashChain@54910693  
Number of ChainLinks in Aggregation Chain[1]:3  
Chain Link[0] Reference:com.guardtime.ksi.unisignature.inmemory.LeftAggregationChainLink@93cb5396  
Chain Link[1] Reference:com.guardtime.ksi.unisignature.inmemory.LeftAggregationChainLink@d15c9d0c  
Chain Link[2] Reference:com.guardtime.ksi.unisignature.inmemory.RightAggregationChainLink@292bc4f0

Hash of Aggregation Chain[2]SHA-256: [B56BF2C17A575D2997B9CFF40D8C7F884C0DD8FA5F676B9A1F793C4D61FAF882]  
Aggregation Chain Reference:com.guardtime.ksi.unisignature.inmemory.InMemoryAggregationHashChain@ad10395b  
Number of ChainLinks in Aggregation Chain[2]:3  
Chain Link[0] Reference:com.guardtime.ksi.unisignature.inmemory.LeftAggregationChainLink@42e14227  
Chain Link[1] Reference:com.guardtime.ksi.unisignature.inmemory.LeftAggregationChainLink@a1b71791  
Chain Link[2] Reference:com.guardtime.ksi.unisignature.inmemory.LeftAggregationChainLink@955bbf6c

Hash of Aggregation Chain[3]SHA-256: [E41C6CE67CD812FD2ED464706BF524559B027D912A331E1C78EC8DBF8949EA3C]  
Aggregation Chain Reference:com.guardtime.ksi.unisignature.inmemory.InMemoryAggregationHashChain@66073029  
Number of ChainLinks in Aggregation Chain[3]:4  
Chain Link[0] Reference:com.guardtime.ksi.unisignature.inmemory.LeftAggregationChainLink@5fddfc0  
Chain Link[1] Reference:com.guardtime.ksi.unisignature.inmemory.LeftAggregationChainLink@1221961f  
Chain Link[2] Reference:com.guardtime.ksi.unisignature.inmemory.RightAggregationChainLink@6123e75e  
Chain Link[3] Reference:com.guardtime.ksi.unisignature.inmemory.RightAggregationChainLink@bad54cf7

Hash of Aggregation Chain[4]SHA-256: [EBA64D3B78838A05FFCEA4B2AC4FB163DF7E269F1D886691B5F649020FDB4EDE]  
Aggregation Chain Reference:com.guardtime.ksi.unisignature.inmemory.InMemoryAggregationHashChain@f7f8f7c0  
Number of ChainLinks in Aggregation Chain[4]:3  
Chain Link[0] Reference:com.guardtime.ksi.unisignature.inmemory.RightAggregationChainLink@aa9d6a5a  
Chain Link[1] Reference:com.guardtime.ksi.unisignature.inmemory.LeftAggregationChainLink@60c9a511  
Chain Link[2] Reference:com.guardtime.ksi.unisignature.inmemory.RightAggregationChainLink@2423b109

Aggregation Time:Tue Oct 18 10:52:06 IST 2016  
Publication Time:Tue Oct 18 10:52:06 IST 2016  
Calendar Hash ChainLinks Size:15  
Calendar Hash Chain Links:  
Calendar Chain[0] Reference:com.guardtime.ksi.unisignature.inmemory.RightInMemoryCalendarHashChainLink@b23e7944  
Calendar Chain[1] Reference:com.guardtime.ksi.unisignature.inmemory.RightInMemoryCalendarHashChainLink@d845ff9b  
Calendar Chain[2] Reference:com.guardtime.ksi.unisignature.inmemory.RightInMemoryCalendarHashChainLink@64ed9ff2  
Calendar Chain[3] Reference:com.guardtime.ksi.unisignature.inmemory.RightInMemoryCalendarHashChainLink@308daaca  
Calendar Chain[4] Reference:com.guardtime.ksi.unisignature.inmemory.RightInMemoryCalendarHashChainLink@61e6aa86  
Calendar Chain[5] Reference:com.guardtime.ksi.unisignature.inmemory.RightInMemoryCalendarHashChainLink@4df72eef  
Calendar Chain[6] Reference:com.guardtime.ksi.unisignature.inmemory.RightInMemoryCalendarHashChainLink@38c8a9c6  
Calendar Chain[7] Reference:com.guardtime.ksi.unisignature.inmemory.RightInMemoryCalendarHashChainLink@2ec2e947  
Calendar Chain[8] Reference:com.guardtime.ksi.unisignature.inmemory.RightInMemoryCalendarHashChainLink@531f5389  
Calendar Chain[9] Reference:com.guardtime.ksi.unisignature.inmemory.RightInMemoryCalendarHashChainLink@d2d749e1  
Calendar Chain[10] Reference:com.guardtime.ksi.unisignature.inmemory.RightInMemoryCalendarHashChainLink@60eb7304  
Calendar Chain[11] Reference:com.guardtime.ksi.unisignature.inmemory.RightInMemoryCalendarHashChainLink@45ae24e  
Calendar Chain[12] Reference:com.guardtime.ksi.unisignature.inmemory.RightInMemoryCalendarHashChainLink@423f949  
Calendar Chain[13] Reference:com.guardtime.ksi.unisignature.inmemory.RightInMemoryCalendarHashChainLink@9f716fb2  
Calendar Chain[14] Reference:com.guardtime.ksi.unisignature.inmemory.RightInMemoryCalendarHashChainLink@67f59dd4  
Signature Data Certificate ID:[B@6b2fad11]  
Signature Data Certificate Repo URI:null  
Signature Data Signature Type:1.2.840.113549.1.1.11  
Signature Data Signature:[B@79698539]  
Publication String:AAAAAAA-CYAWYX-4AOWUC-BNYPHW-OATJSO-0THP2X-DS4PYB-SRTL4G-3LSRMM-F55204-IVQ7ZM-IXGNFH  
Publication Data Hash:SHA-256: [D6A082DC3CF670269939D33BF571CB8FC06519AF86DAE51630BDEE9DC4561FCB]  
Publication Time:Tue Oct 18 10:52:06 IST 2016