Fast and Secure Handoffs for V2I Communication in Smart City Wi-Fi Deployment

Pranav Kumar Singh(✉), Subhredu Chattopadhyay, Pradeepkumar Bhale, and Sukumar Nandi

Department of Computer Science and Engineering, Indian Institute of Technology, Guwahati 781039, India
{pranav.singh,subhrendu,pradeepkumar,sukumar}@iitg.ernet.in

Abstract. The Intelligent Transport System (ITS) is a vital part of smart city developments. Due to densely deployed access points and vehicular mobility in a smart city, the number of handovers also increases proportionately. Minimization of the handoff latency is crucial to provide a better quality of service for vehicles to have access different ITS services and applications. Increased handover latency can cause an interruption in vehicle-to-infrastructure (V2I) communication. In this paper, we propose a fast and secure handoff mechanism for smart cities that have acceptable handoff latency for delay-sensitive ITS applications and services. Our proposal considers mobility and communication overhead to provide lower handoff latency. We compare our proposed mobility aware background scanning mechanism (AdBack) with standard Active Scanning mechanism in an emulated test bed. Our test results reveal that the proposed AdBack mechanism significantly outperforms the existing mechanisms in terms of handover latency, packet drop rates, and throughput. Experimental results show that amalgamation of AdBack and existing fast re-authentication (IEEE 802.11r) can improve connectivity for V2I communication in a smart city. We provide rigorous emulation results to justify the performance of our proposed scheme.

1 Introduction

Vehicular communication that emerged from wireless communication has gained much interest from academia, industries, and governments to improve road safety, fuel efficiency, and convenience of travel. Vehicular communication is one of the leading research areas because of its applications and its specific characteristics. In the wake of the Information and communication technology (ICT) revolutions, road transportation has entered into a new era, and today we have technologies and services such as connected vehicles, driverless cars, smart cars, VANET, Internet of vehicles, vehicle telematics, and intelligent transportation systems (ITS). Based on the data collected from various studies, surveys, polls and driver's experiences hundreds of applications can be suggested. Most of

Aloe: An Elastic Auto-Scaled and Self-stabilized Orchestration Framework for IoT Applications

Subhrendu Chattopadhyay*, Soumyajit Chatterjee†, Sukumar Nandi‡, Sandip Chakraborty§
*‡ Department of CSE, IIT Guwahati, India
†§ Department of CSE, IIT Kharagpur, India

***Abstract*—Management of networked Internet of Things (IoT) infrastructure with in-network processing capabilities is becoming increasingly difficult due to the volatility of the system with low-cost resource-constraint devices. Traditional software-defined networking (SDN) based management systems are not suitable to handle the plug and play nature of such systems. Therefore, in this paper, we propose Aloe, an elastically auto-scalable SDN orchestration framework. Instead of using service grade SDN controller applications, Aloe uses multiple lightweight controller instances to exploit the capabilities of in-network processing infrastructure. The proposed framework ensures the availability and significant reduction in flow-setup delay by deploying instances near the resource constraint IoT devices dynamically. Aloe supports fault-tolerance and can recover from network partitioning by employing self-stabilizing placement of migration capable controller instances. The performance of the proposed system is measured by using an in-house testbed along with a large scale deployment in Amazon web services (AWS) cloud platform. The experimental results from these two testbed show significant improvement in response time for standard IoT based services. This improvement of performance is due to the reduction in flow-setup time. We found that Aloe can improve flow-setup time by around $10\% - 30\%$ in comparison to one of the state of the art orchestration framework.**

***Index Terms*—Network architecture, Fault-tolerance, Programmable network, software defined network**

I. Introduction

The rapid proliferation of Internet-of-Things (IoT) has made the network architecture complicated and difficult to manage for service provisioning and ensuring security to the end-users. Simultaneously, with the advancement of edge-computing, in-network processing (also known as fog computing) and platform-as-a-service technologies, end-users consider the network as a service platform for the deployment and execution of myriads of diverse applications dynamically and seamlessly over the network. Consequently, network management is becoming increasingly difficult in today's world with this complex service-oriented platform overlay on top of the inherently distributed TCP/IP network architecture. The concept of software-defined networking (SDN) has gained popularity over the last decade to make the network management simple, cost-effective and logically centralized, where a network manager can monitor, control and deploy new network services through a central controller. Nevertheless, edge and in-network processing over an IoT platform is still challenging even with an SDN based architecture [1].

The primary requirements for supporting edge and in-network processing over a networked IoT platform are as follows: (1) The platform should be agile to support rapid deployment of applications without incurring additional overhead for in-network processing [2]. This also ensures scalability of the system [3]. (2) Many times, in-network processing requires dividing a service into multiple microservices and deploying the microservices at different network nodes for reducing the application response time with parallel computations [4]. However, such microservices may need to communicate with each other, and therefore the flow-setup delay from the in-network nodes need to be very low to ensure near real-time processing. (3) The percentage of short-lived flows are high for IoT based networks [5]. This also escalates the requirement for reducing flow-setup delay in the network. (4) Failure rates of IoT nodes are in-general high [6]. Therefore, the system should support a fault-tolerant or fault-resilient architecture to ensure liveness.

Although SDN supported edge computing and in-network processing have been widely studied in the literature for the last few years [1], [7] as a promising technology to solve many of the network management problems associated with large-scale IoT networks, they have certain limitations. First of all, the SDN controller is a single-point bottleneck. Every flow initiation requires communication between switches and the controller; therefore, the performance depends on the switch-controller delay. With a single controller bottleneck, the delay between the switch and the controller increases, which affects the flow-setup performance. As we mentioned earlier that the majority of the flows in an IoT network are short-lived flows, the impact of switch-controller delay is more severe on the performance of short-lived flows. To solve this issue, researchers have explored distributed SDN architecture with multiple controllers deployed over the network [8]. However, with a distributed SDN architecture, the question arises about how many controllers to deploy and where to deploy those controllers. Static controller deployments may not alleviate this problem, as IoT networks are mostly dynamic with a plug-and-play deployment of devices. Dynamic controller deployment requires hosting the controller software over commercially-off-the-shelf (COTS) devices and designing methodologies for

Email:subhrendu@iitg.ac.in*,soumyachat@iitkgp.ac.in†,sukumar@iitg.ac.in‡, sandipc@cse.iitkgp.ac.in§

This work is partially supported by TCS India, Microsoft India, LRN Foundation and CNeRG IIT Kharagpur.

Amalgam: Distributed Network Control With Scalable Service Chaining

Subhrendu Chattopadhyay
IIT Guwahati
Guwahati, India 781039
subhrendu@iitg.ac.in

Sukumar Nandi
IIT Guwahati
Guwahati, India 781039
sukumar@iitg.ac.in

Sandip Chakraborty
IIT Kharagpur
Kharagpur, India 721302
sandipc@cse.iitkgp.ernet.in

Abhinandan S. Prasad
NIE Mysuru
Karnataka, India 570008
abhinandansp@nie.ac.in

***Abstract*—Management of virtual network function (VNF) service chaining for a large scale network spanning across multiple administrative domains is difficult due to the decentralized nature of the underlying system. Existing session-based and software-defined networking (SDN) oriented approaches to manage service function chains (SFCs) fall short to cater to the plug-and-play nature of the constituent devices of a large scale eco-system such as the Internet of Things (IoT). In this paper, we propose *Amalgam*, a composition of a distributed SDN control plane along with a distributed SFC manager, that is capable of managing SFCs dynamically by exploiting the in-network processing platform composed of plug-and-play devices. To ensure the distributed placement of VNFs in the in-network processing platform, we propose a greedy heuristic. Further, to test the performance, we develop a complete container driven emulation framework *MiniDockNet* on top of standard *Mininet* APIs. Our experiments on a large scale realistic topology reveal that *Amalgam* significantly reduces flow-setup time and exhibits better performance in terms of end-to-end delay for short flows.**

***Index Terms*—Service function chaining, Virtual network function, In-network processing, Programmable network, software defined network**

I. Introduction and Related Works

Due to the rapid deployments of connected environments, large-scale Internet of Things (IoT) networks [1] have become prevalent in recent years. Management of such large-scale heterogeneous ecosystems requires various network services such as network address translator (NAT), firewall, proxy, and local domain name server (DNS); these network services are called network function (NF)s. Generally, the network functions are deployed using virtual machines (VM)(s) to provide service isolation and reduce CapEx and OpEx; therefore, they are termed as virtualized network function (VNF) [1]. VNFs execution require computation platform to host the VM and execute the NF within the VM. Depending on network management policies, the application messages require steering through an ordered set of VNFs known as service function chaining (SFC) [2].

Among various existing architectures to execute VNFs over a network infrastructure [3]–[5] relies on software-defined network (SDN) [6] to steer flows from one VNF to another. On the other hand, [7], [8] takes a session-based approach where the end hosts control the SFC. Session-based approaches achieve lower host-based state management of VNFs, where SDN-based approaches achieve fine-grained quality of service (QoS). However, for a large-scale network spanning across multiple administrative domains, both of the SFC management (SCM) approaches fall short in several aspects as follows.

(a) Lack of scalability: Existing SCMs [6], [9] use a central controller that monitors the resource usage of the devices and use as the basis for the VNFs deployment. The use of a central controller for VNF deployment becomes challenging, especially when the network spans across multiple autonomous administrative domains that interconnected through different network service providers. On the other hand, the VNF placement is $\mathcal{NP}$-hard [10]. Existing distributed heuristics for VNF placement [11] require multiple rounds to deploy VNFs, which increases flow initiation delay leading to reduced IoT application performance since the majority of the IoT flows are short-lived [12].

(b) Dynamic service chaining: Usually, VNFss modifying the headers are common in a large-scale network. Consequently, the participating VNFs can change the SFCs during the lifetime of a flow based on the flow characteristics. For instance, a classifier VNF can add a load balancer based on the arrival rate of the packets in a flow. Existing scalable distributed VNF placement methods [11] and IP based traffic steering proposals [13] are not suitable for dynamic service chaining. On the other hand, [7] ascertains dynamic service chaining by adding an agent in each device, including hosts. Installation of agents on a large scale IoT becomes infeasible, where the devices with plug-and-play capability can dynamically enter and exit the ecosystem.

(c) Issues of flow monitoring over multi-administrative platforms: To steer the traffic through proper service chains while ensuring QoS, requires fine-grained flow monitoring. Existing flow identification methods using packet header fields are insufficient in the presence of a header modifying VNF in the SFC (such as NAT, load balancer, and proxy). Existing SDN-based flow monitoring schemes like FlowTags [9], Stratos [14] utilize *"vlan/mpls"* tagging which does not work through multiple administrative domains. On the other hand, the use of packet encapsulation in session-based approaches

This work was supported by TCS India Pvt. Ltd.

[1]https://www.sigfoxcanada.com/

Annex to ISBN 978-3-903176-28-7 © 2020 IFIP

DisProTrack: Distributed Provenance Tracking over Serverless Applications

Utkalika Satapathy*, Rishabh Thakur*, Subhrendu Chattopadhyay†, Sandip Chakraborty*
*IIT Kharagpur, Kharagpur, India 721302 †IDRBT, Hyderabad, India 500057

***Abstract*—Provenance tracking has been widely used in the recent literature to debug system vulnerabilities and find the root causes behind faults, errors, or crashes over a running system. However, the existing approaches primarily developed graph-based models for provenance tracking over monolithic applications running directly over the operating system kernel. In contrast, the modern DevOps-based service-oriented architecture relies on distributed platforms, like serverless computing that uses container-based sandboxing over the kernel. Provenance tracking over such a distributed micro-service architecture is challenging, as the application and system logs are generated asynchronously and follow heterogeneous nomenclature and logging formats. This paper develops a novel approach to combining system and micro-services logs together to generate a Universal Provenance Graph (UPG) that can be used for provenance tracking over serverless architecture. We develop a Loadable Kernel Module (LKM) for runtime unit identification over the logs by intercepting the system calls with the help from the control flow graphs over the static application binaries. Finally, we design a regular expression-based log optimization method for reverse query parsing over the generated UPG. A thorough evaluation of the proposed UPG model with different benchmarked serverless applications shows the system's effectiveness.**

***Index Terms*—Distributed provenance tracking,**

I. Introduction

Modern service-oriented architecture adopts DevOps [1], [2] practices and technologies to provide Software as a service (SaaS) [3] by leveraging distributed cloud infrastructure. Service deployment on top of the cloud widely adopts serverless computing (SLC) [4] to reduce operational expenditure whenever the service computations are stateless, elastic, and possibly distributed. Micro-services deployed on top of SLC architecture provide an abstraction of the underlying infrastructure where the developer can write, deploy and execute the code without configuring and managing the shared environment [4]–[7]. However, the available serverless-specific industry solutions [8]–[10] provide limited support for error reporting, execution tracing, and provenance tracking. Consequently, developers can only provide little attention to log vital forensic information. Some of the third-party observability tools [11]–[14] support distributed tracing as well as cost analysis features by instrumenting the source codes. However, these tools only support applications developed using a particular programming language. So, it is difficult to analyze the actual behavior of these micro-services.

Provenance Graphs: The non-invasive [1] frameworks rely on the logs generated by applications to identify the execution states. Since most of the production-grade micro-services are chosen from an available stable release, the executable files already contain meaningful log messages that can be used to identify event handling loops. In the domain of system security, provenance data is the metadata of a process that records the details of the origin and the history of modification or transformation that happened over time throughout its lifecycle [15]–[20]. The graph generated from this information is called the *provenance graph* of a process which is a causal graph that stores the dependencies between system subjects (e.g., processes) and system objects (e.g., files, network sockets). For example, an application event may generate a separate application log, error log, and operating system calls specific log, etc. In this context, a provenance graph is a directed acyclic graph (DAG) where individual log entries are the nodes, and the edges represent the causality relationship between the log entries. During attack investigation, an administrator queries this graph to find out the root cause and ramifications of an event. While a malicious entity performs some illegal events, the corresponding system logs are recorded as the provenance data. For example, when a compromised process tries to open a sensitive file, the OS level provenance can record the file-open activity, which can be referred for vulnerability analysis whenever an attack is detected in the system [21], [22]. Real-world enterprises widely use kernel or OS level information (logs) to perform provenance analysis to monitor their systems and identify the malicious events performed back in time.

A. Limitations of Existing Works and the Research Challenges

There have been several works that attempted to generate the provenance graphs by combining system and application logs together [16]–[20], [23]. However, these works primarily considered a monolithic application running directly over the Kernel, and thus combining the system log with the application log is not difficult. In contrast, the individual log files for each micro-service over an SLC application are physically distributed across the entire eco-system, which makes the generation of the provenance graph non-trivial. In such a scenario, a Universal Provenance Graph (UPG) that combines the interactions among all the micro-services can provide a meaningful platform for distributed provenance tracking. A few existing works [20], [24] have tried to address the issue of encoding all forensically-relevant causal dependencies regardless of their origin. Nevertheless, we have some non-trivial

[1]The non-invasive tools neither inject any piece of code inside the micro-service/container instances nor injects any special services into the SLC platform.

Surpassing Flow Fairness in a Mesh Network: How to Ensure Equity among End Users?

Sandip Chakraborty, Sukumar Nandi, Subhrendu Chattopadhyay
Department of Computer Science and Engineering,
Indian Institute of Technology Guwahati, Assam, India 781039
Email: {c.sandip, sukumar, subhrendu}@iitg.ernet.in

***Abstract*—Assuring user fairness in IEEE 802.11 wireless mesh network is challenging mainly for two reasons - first, the in-built unfair behavior of IEEE 802.11 contention based channel access in multi-hop forwarding, and second, the capacity degradation of mesh routers near a gateway due to the heavy traffic load. This paper studies the fundamental reason of fairness degradation in a mesh network, using a practical mesh testbed. The experimental results reveal that the flow fairness alone cannot solve this issue, because the near-gateway routers with high traffic load require more bandwidth for traffic forwarding. An improved channel access differentiation technique based on a load estimation strategy is proposed in this paper to reduce the capacity degradation of mesh routers, and to improve the fairness among end users. The proposed protocol is evaluated using results obtained from two different testbed setups. These results show that the proposed protocol improves the end-user fairness, as well as the average throughput in the network.**

***Keywords*-mesh; IEEE 802.11; fairness; capacity; flow control**

I. Introduction

IEEE 802.11 Wireless mesh network [1] has generated interests among researchers because of its capability of replacing the wired infrastructure by wireless backbone. Mesh backbone network is mainly comprised of wireless routers which have two functionalities - to act as the access point (AP) to client nodes, and to relay the traffic from other routers towards or from the gateway. One or more of the mesh routers also act as gateways to connect outside network.

A mesh network over IEEE 802.11 technology experiences severe problem of unfairness among end-users. IEEE 802.11 Enhanced Distributed Channel Access (EDCA) provides equal time share to every contending station when every station has sufficient data to transmit. Therefore, mesh routers with high traffic load (mainly routers near a gateway) suffer from the unavailability of sufficient bandwidth, and the performance degrades drastically for the end users who are multiple hops away. Though several proposals exist in literature to study fairness issues in multi-hop and mesh networks, such as [2]–[4] and their references, they primarily concentrate on end-to-end flow fairness. A second class of works studies fairness issues in channel access protocols, such as [5]–[7] and their references, that mainly focus on prioritizing the channel access of individual users, according to their traffic demand and the quality of service criteria. However, these works do not reveal the fundamental reasons behind the fairness degradation in a mesh network. Assuring end-to-end fairness cannot solve this issue alone, as the capacity of mesh routers near a gateway degrade severely at high traffic load. Therefore traffic control at intermediate routers is necessary to ensure equal transmission opportunity to every end-user.

In our previous paper [8], an improved fairness provisioning protocol has been proposed that estimates traffic load at every intermediate router. The channel access mechanism of every router is either prioritized or deprived based on the traffic load and the contention information in the neighborhood. Therefore, every router gets channel access proportional to its traffic load (called the *proportional fairness* criteria) that ensures traffic equality among all flows. The contention window (CW) of IEEE 802.11 EDCA has been tuned to prioritize or to deprive the channel access of individual routers. However, the load estimation strategy proposed earlier is based on the traffic demand of individual clients that varies with time according to application usage, and thus difficult to compute, account and manage at router level. Further the CW tuning mechanism considers only short-term effect that may result in sudden performance degradation when new flows join in the network, or existing flows terminate.

The objective of this paper is two-fold. The first objective is to study the unfair behavior of end-users in a mesh network even with the support of an end-to-end flow-fairness control protocol, such as fair queuing, and second the second objective is to provide an optimized implementation of the previously proposed protocol [8] with an improved efficiency. A practical testbed of mesh network has been used for this purpose. The major contributions of this paper are summarized as follows.

- The results obtained from a practical testbed are used to study the problem of fairness degradation in a mesh network even with the support of end-to-end flow fairness at transport layer. The study reveals that clients associated with the routers near the gateway reserve the full capacity of the network, as a consequence of which, the flows with large number of hops get starved.
- An improved protocol is proposed over [8] to support channel access based on the traffic load at individual routers. The load estimation strategy only relies on the router level information, such as the size of the interface queue, and therefore free from the time varying information, that may introduce stale results. Further, the CW tuning mechanism is optimized by considering long-term

A Trigger Counting Mechanism for Ring Topology

Sushanta Karmakar[1] **Subhrendu Chattopadhyay** [1]

[1] Department of Computer Science and Engineering,
Indian Institute of Technology Guwahati, India, 781039
Email: sushantak@iitg.ernet.in, subhrendu@iitg.ernet.in

.

Abstract

Consider a distributed system with n processors, which receive triggers from the outside world. The Distributed Trigger Counting (DTC) problem is to raise an alarm if the number of triggers over the system reaches w, which is an user specified input. DTC is used as a primitive operation in many applications, such as distributed monitoring, global snapshot etc. In this paper, we propose an algorithm for the DTC problem in a ring topology with a message complexity of $O(n^2 \log(w/n))$ and each node in the system receives $O(n \log(w/n))$ number of messages. We also discuss about the possible tuning of the algorithm which results better complexities.

Keywords: Distributed algorithm, distributed monitoring, distributed trigger counting, ring topology

1 Introduction

Distributed trigger counting (DTC) is an important problem in distributed systems. Consider a distributed system with n processes. Each process receives some triggers (signals) from an external source. The DTC problem is to detect the state when the number of triggers received by the system reaches w which is a user defined input to the system. Note here that w may be much larger than n. The sequence of processors receiving the w triggers is not known apriori to the distributed system. Our goal is to propose an algorithm for the DTC problem under a known topological setting. More specifically, in this paper we propose an algorithm for the DTC problem in a ring network.

The DTC problem arises in many applications in distributed systems. Some major application areas can be distributed monitoring, global snapshot etc. Monitoring is an important aspect in wireless networks as well as in wired networks. A wireless sensor network is typically used to monitor physical or environmental conditions such as border surveillance, forest fire detection, traffic management in highways etc. In traffic management, one may be interested in detecting whether the number of vehicles on a highway exceeds a certain threshold. Similar applications may be found in border surveillance and forest fire detection as well. In distributed system a global snapshot state is said to be valid provided all the in-transit messages are recorded. It was shown by Garg et al. (2006) that the problem of detecting whether all the in-transit messages have been recorded can be reduced to the DTC problem. Awerbuch (1985) proposed the concept of synchronizers which is a tool to transform a synchronous distributed algorithm to another that runs on an asynchronous distributed system. Here a distributed system is required to generate a signal (or pulse) when all the messages generated after the previous signal have been delivered. This problem can also be formulated as a variant of the DTC problem. The performance of an algorithm for the DTC problem is often measured by the following two metrics.

- *Message complexity:* It is the total number of messages sent by all the nodes of the systems.
- *MaxRcvLoad:* It is the maximum number of messages received by any node in the system.

The DTC problem was studied by Garg et al. (2006) for a general distributed system. In their work, they proposed two algorithms for the DTC problem: a centralized algorithm and a tree-based algorithm. The centralized algorithm has a message complexity of $O(n \log w)$. Its MaxRcvLoad has a complexity of $O(n \log w)$. This is a tight bound for the centralized algorithm. The tree based algorithm has a message complexity of $O(n \log n \log w)$. Again the MaxRcvLoad of the tree based algorithm has a complexity of $O(n \log n \log w)$. In the work the authors also showed that any deterministic algorithm for the DTC problem must have a message complexity of $\Omega(n \log(w/n))$. Huang et al. (2007) proposed a novel solution to the problem of efficient detection of an aggregate predicate over cumulative triggers over a time-varying window in a distributed monitoring system. They provided a queueing theory based analysis of the problem which provides the user the power to trade-off between desired accuracy and communication overhead. However, their approach is based on a single coordinator. Hence the algorithm is not fully distributed. In another work (Chakaravarthy, Choudhury, Garg & Sabharwal 2011) a randomized distributed algorithm was proposed for trigger counting in a tree based topology. The algorithm has a message complexity of $O(n \log n \log w)$ and MaxRcvLoad complexity of $O(\log n \log w)$ with high probability. Later in another work (Chakaravarthy, Choudhury & Sabharwal 2011) the authors proposed an improved algorithm for the DTC problem having message complexity of $O(n \log w)$ and MaxRcvLoad of $O(\log w)$. However, this work has the limitation that the algorithm is a randomized algorithm and it does not provide any bound on the messages sent per node.

Defending Concealedness in IEEE 802.11n

Sandip Chakraborty, Subhrendu Chattopadhyay, Suchetana Chakraborty, Sukumar Nandi
Department of Computer Science and Engineering, IIT Guwahati, Assam, India 781039
Email: {c.sandip, subhrendu, suchetana, sukumar}@iitg.ernet.in

***Abstract*—IEEE 802.11 distributed coordination function supports two access mechanisms - the basic access and the four-way access, both of which are vulnerable to the hidden and the exposed node problem (collectively called the concealed node problem). Though most of the works in literature suggest to overlook this problem due to the associated overhead to solve them, this paper shows that the problem becomes severe for high speed wireless mesh networks. An opportunistic four-way access mechanism is designed to defend this problem in IEEE 802.11n mesh networks that supports high data rates. The performance of the proposed scheme is evaluated in a practical 802.11n indoor mesh testbed, that shows a significant performance improvement compared to the standard access mechanisms.**

***Keywords*-IEEE 802.11n; exposed; opportunistic access**

I. Introduction

IEEE 802.11 *'Carrier Sense Multiple Access with Collision Avoidance'* (CSMA/CA) supports two different access technologies - the *basic access*, where every data frame is followed by a corresponding acknowledgement (ACK) frame, and the *four-way access*, where two control frames are used to reserve channel before the actual data communication, namely the 'Request to Send' (RTS) and the 'Clear to Send' (CTS) frames. The basic access mechanism results in the well known *hidden node* problem [1]. Considering Fig. 1, both the nodes A and C want to transmit to the node B simultaneously. The basic access mechanism uses the concept of physical carrier sensing (PCS) before the data communication, where the nodes sense the channel for being idle. As node C is outside the carrier sense (CS) range of node A, it can not sense the ongoing data transmissions $A \to B$. As a result, node C may initiate another communication with node B, resulting interference near the receiver. The four-way access mechanism [2] solves the hidden node problem by communicating with RTS and CTS handshaking control frames before the actual data transmission. On overhearing these control frames, every node in the CS region of the transmitter and the receiver defer its transmissions to avoid interference. This procedure is called virtual carrier sensing (VCS). However, the four way access mechanism introduces another problem, called the *exposed node problem* [3]. Considering Fig. 1, a node G, outside the CS range of the receiver node B, can initiate communication from node F. However, on overhearing the CTS frame from node

The works of Sandip Chakraborty and Suchetana Chakraborty are supported by the TATA Consultancy Services (TCS), India through the TCS Research Fellowship Program. The authors would like to thank Mr. Rajendra Singh, Skiva Technologies for providing necessary hardware and technical supports to implement the testbed.

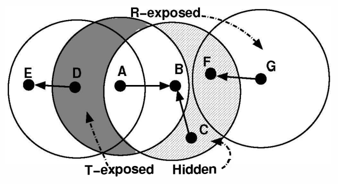

Fig. 1. Hidden, T-exposed and R-exposed nodes

B, node F defers all the communication through it. Therefore, on receiving an RTS frame from node G, node F does not reply back with a CTS frame to initiate the communication. Here, F is a receiver-side exposed node, termed as *R-exposed* node in this paper, that can act as a potential receiver, but is blocked due to the VCS. Similarly, from Fig. 1, exposed node problem can also occur at the transmitter side because of the communication blockage due to RTS overhearing. These nodes are termed as *T-exposed* in this paper. In the figure, D is a T-exposed node that can also act as a potential transmitter for another receiver E, outside the CS range of the node B.

The effects of the concealed nodes for IEEE 802.11 basic and four-way access mechanisms have been well studied in literature, such as [4] and the references therein. Recent researches have shown that RTS/CTS access mechanism does not perform well always, because of the signaling overhead [5] due to control message transmission. Nevertheless, the basic access mechanism suffers from both the hidden and the T-exposed nodes. On the contrary, the four-way access can solve the hidden node problem, but undergoes for both the T-exposed and R-exposed nodes. Though the exposed nodes reduce the spatial reuse, allowing communications to these nodes may result in data-ACK interference, if not properly synchronized. However, as shown in this paper, the exposed node problem becomes severe in the case of high data rate mesh networks built upon the 802.11n technology. In [6], the authors have shown that RTS/CTS exchange can results in performance drop for high data rate mesh networks by increasing number of exposed nodes. However, they have not considered the complete features of 802.11n high speed networks, such as frame aggregation and block ACK (BACK). A number of works exist for mitigating the exposed terminal problem, such as [3] and the references therein, however, they require even additional messages over RTS/CTS, and do not consider the advanced technologies supported by 802.11n. This paper theoretically models the performance of the high data rate mesh networks for three different scenarios with concealed nodes to show their effect over the high speed network

Leveraging the Trade-off Between Spatial Reuse and Channel Contention in Wireless Mesh Networks

Subhrendu Chattopadhyay
IIT Guwahati
Guwahati, India 781039
subhrendu@iitg.ernet.in

Sandip Chakraborty
IIT Kharagpur
Kharagpur, India 721302
sandipc@cse.iitkgp.ernet.in

Sukumar Nandi
IIT Guwahati
Guwahati, India 781039
sukumar@iitg.ernet.in

***Abstract*—Performance optimization strategies in wireless mesh networks have witnessed many diverged directions, out of which the recent studies have explored the use of spatial multiplexing through data rate and transmit power adaptation to increase spatial re-usability while minimizing network interference. However, joint data rate and transmit power adaptation shows a clear trade-off, where higher transmit power helps in sustaining high achievable data rates with the cost of increased interference to the neighboring receiver nodes. Such a trade-off results in channel access unfairness among contending flows, where a node with high transmit power gains more performance benefit compared to its neighbors. Unfairness among nodes in a mesh network results in performance drops for the end-to-end flows. In this paper, we formulate a multivariate optimization problem to maximize network utilization and fairness, while minimizing average transmit power for all transmitter nodes. The Pareto optimality nature of the vector optimization has been explored to design a distributed localized heuristic where every node individually decides their scheduling slots and transmit power while keeping fairness as a constraint. The performance of the proposed scheme has been evaluated through simulation, and the comparisons with recent studies reveal that it improves fairness that results approximately $10\% - 40\%$ improvement in end-to-end throughput for different network and traffic scenarios.**

***Keywords*-Fairness; Power adaptation; Rate Control; Scheduling; Joint Optimization; MCCA**

I. Introduction

Wireless mesh network (WMN) is one of the promising technologies for providing cost efficient and robust backbone infrastructure for broadband connection over metropolitan area networks. IEEE 802.11 task group for wireless local area network provides the standard IEEE 802.11s [1], that defines the media access control (MAC) layer channel access and forwarding for multi-hop wireless mesh networks. IEEE 802.11s supports mesh coordination function (MCF) for supporting quality of service (QoS) provisioning in mesh functionalities, which defines MCF controlled channel access (MCCA) for MAC layer scheduling of mesh nodes.

The design and performance of MAC layer protocols for a multi-hop mesh networks significantly depend on several physical layer parameters - like modulation and coding schemes (MCS), transmit powers, channel widths etc, which are further interdependent. Higher MCS levels provide better physical data rate, however require higher transmit power for sustainability. If a transmitter uses higher transmit power level, the data decoding probability at the receiver increases by supporting higher signal to interference and noise ratio (SINR). This can be realized by the following relation between transmit power, physical data rate and SINR [2];

$$SINR = \frac{P_{ij}G_{ij}}{\eta + \sum_{\substack{k \in I \\ k \neq i \\ x \neq j}} G_{kj}P_{kx}} \geq \gamma(r_h) \tag{1}$$

Here P_{ij} and G_{ij} denote the transmit power level and "antenna and channel gain" for a wireless link between transmitter i to receiver j, respectively. I is the set of nodes in the network. η is the ambient noise and $\gamma(r_h)$ is the receive sensitivity for data rate r_h, where r_h corresponds to the physical data rate for MCS level h. If the transmission power is increased, the other receiver nodes in the communication range of the transmitter experience interference, that reduces spatial re-usability of the available channel space. From Eq. (1), it is also evident that the physical data rate has also a direct relationship with the transmit power in forms of receive sensitivity. Receive sensitivity can be defined as the required SINR for correct decoding of a frame transmitted in a particular data rate, that withstands a standard bit error rate (BER) [3]. If SINR of a received frame is greater than the receive sensitivity, then only a node can decode the frame successfully. It has been observed that higher the data rate, greater the receive sensitivity. Notationally for different data rates $r_1 < r_2 < ... < r_m$, $\gamma(r_1) < \gamma(r_2) < ... < \gamma(r_m)$. In summary, the trade-off between data rate and transmit power with respect to spatial re-usability and interference can be captured as follows:

(i) Higher data rates require higher receiver sensitivity, which implies that they can sustain only at higher SINR levels.
(ii) Higher transmit powers provide higher SINR, however in the cost of increased interference and reduced spatial re-usability of the channel.

Therefore, choosing a proper scheduling of transmit power level along with proper MCS level to gain the optimum throughput is a challenge in case of WMN. In a recent study, Li *et.al.* [4] have shown that optimal power control strategy with scheduling even for only two discrete power levels is $\mathcal{NP}$-hard because of the additive/cumulative interference effect on the receiver node.

2016 8th International Conference on Communication Systems and Networks (COMSNETS)

ES2: Managing Link Level Parameters for Elevating Data Rate and Stability in High Throughput WLAN

Sandip Chakraborty
Department of Computer Science and Engineering,
Indian Institute of Technology Kharagpur,
Kharagpur, India 721302
Email: sandipc@cse.iitkgp.ernet.in

Subhrendu Chattopadhyay
Department of Computer Science and Engineering,
Indian Institute of Technology Guwahati,
Guwahati, India 781039
Email: subhrendu@iitg.ernet.in

***Abstract*—High Throughput (HT) IEEE 802.11n/ac wireless networks support a large set of configuration parameters, like Multiple Input Multiple Output (MIMO) streaming, modulation and coding scheme, channel bonding, short guard interval, frame aggregation levels etc., that determine its physical data rate. However, all these parameters have an optimal performance region based on the link quality and external interference. Therefore, dynamically tuning the link parameters based on channel condition can significantly boost up the network performance. The major challenge in adapting all these parameters dynamically is that a large feature set need to be enumerated during run-time to find out the optimal configuration, which is a not feasible in real time. Therefore in this paper, we propose an estimation and sampling mechanism to filter out the non-preferable features on the fly, and then apply a learning mechanism to find out the best features dynamically. We apply a Kalman filtering mechanism to figure out the preferable feature sets from all possible feature combinations. A novel metric has been defined, called the *diffESNR*, which is used to select the best features from the sampled feature sets. The proposed scheme, *Estimate-Sample-Select* (*ES2*) is implemented and tested over a mixed wireless testbed using IEEE 802.11n and IEEE 802.11ac HT wireless routers, and the performance is analyzed and compared with other related mechanisms proposed in the literature. The analysis from the testbed shows that *ES2* results in approx 60% performance improvement compared to the standard and other related mechanisms.**

***Keywords*-High Throughput Wireless; link adaptation; IEEE 802.11n; IEEE 802.11ac**

I. Introduction

The invent and wide-spread deployments of high throughput (HT) wireless technologies, like IEEE 802.11n or IEEE 802.11ax [1]–[3], give a breakthrough to the wireless local area networking (WLAN) for commodity and community wireless usage over free bandwidth. The HT wireless technologies support high physical data rates (IEEE 802.11n supports data rates up to 600 Mbps, whereas IEEE 802.11ax supports rate in Gbps) through a number of physical and media access control (MAC) layer advancements; like the inclusion of multiple antenna technology through *Multiple Input Multiple Output* (MIMO), channel bonding and frame aggregation. The MIMO supports spatial division multiplexing (SDM) with space time block coding (STBC) for improving network capacity. The IEEE 802.11n standard supports 2 spatial multiplexed streams. The channel bonding feature provides wider channels of 40 MHz (for IEEE 802.11n) or 80 MHz (for IEEE 802.11ax) by combining multiple narrow channels of 20 MHz together. Frame aggregation combines multiple upper layer frames, or protocol data unit (PDU), to reduce channel access overhead.

Apart from HT wireless specific features, modern wireless routers also provide different modulation and coding schemes (MCS) that support different data rates. The existing studies [4], [5] have revealed that optimal selection of MCS depends on channel and network conditions, and accordingly different rate adaptation mechanisms have been proposed for legacy WLAN networks. However, HT wireless networks combine a large number of feature set combinations based on the link configuration parameters, like number of spatial streams, channel bondings, guard intervals, MCS levels, frame aggregation levels etc. For example, IEEE 802.11n supports 256 possible combinations of feature sets - 4 spatial streams, 2 channel widths (20 MHz and 40 MHz), 2 guard intervals (800 ns and 400 ns), 8 different MCS levels and 2 frame aggregation levels (ON or OFF). A number of recent studies [5]–[14] have shown that dynamic adaptation from these feature sets provides better network performance compared to static assignments. Accordingly, several link adaptation mechanisms have been proposed in the literature, and Minstrel HT [15] has been widely accepted as the default link adaptation mechanism for HT wireless networks.

The existing link adaptation mechanisms for HT wireless networks can be broadly classified into two groups - open loop link adaptation and closed loop link adaptation. Open loop link adaptation [5]–[7] uses some form of sampling to find out the best set of features from the available feature set. However, the feature sets are extensively large for online processing and therefore, the existing approaches use some kind of filtering on static assignments of few features. The closed loop approaches [8] rely on receiver feedback which incurs significant overhead and processing delay to the network. Consequently, the link adaptation mechanism for HT wireless network remains an open issue to the research community.

In summary, link adaptation in HT wireless networks faces two challenges - first, the number of features in the feature space is very large for on-line processing and optimization, and second, both capacity (or throughput) and fairness need to be

A time aware method for predicting dull nodes and links in evolving networks for data cleaning

Niladri Sett, Subhrendu Chattopadhyay, Sanasam Ranbir Singh, Sukumar Nandi
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati
Guwahati, India-781039
Email: {niladri,subhrendu,ranbir,sukumar}@iitg.ernet.in

***Abstract*—Existing studies on evolution of social network largely focus on addition of new nodes and links in the network. However, as network evolves, existing relationships degrade and break down, and some nodes go to hibernation or decide not to participate in any kind of activities in the network where it belongs. Such nodes and links, which we refer as "dull", may affect analysis and prediction tasks in networks. This paper formally defines the problem of predicting dull nodes and links at an early stage, and proposes a novel time aware method to solve it. Pruning of such nodes and links is framed as "network data cleaning" task. As the definitions of dull node and link are non-trivial and subjective, a novel scheme to label such nodes and links is also proposed here. Experimental results on two real network datasets demonstrate that the proposed method accurately predicts potential dull nodes and links. This paper further experimentally validates the need for data cleaning by investigating its effect on the well-known "link prediction" problem.**

Keywords*—*Social network analysis, Dull node, Dull link, Time-series, Link prediction.

I. Introduction

Evolution of social network [1], [2] has been receiving substantial attention in the field of social network analysis (SNA). A social *tie* (or a link) is built through social interaction between two *actors* (or nodes); and structure of a social network evolves with addition of new nodes and links in the network. Association or friendship between two actors is maintained by the amount of information that flows between them, which is often quantified using the pattern of interactions they are involved in. Most of the studies on the evolution of social network have concentrated on addition of new links and nodes in it. However, in reality, some links become inactive with time as friendships break down, and some nodes withdraw themselves from the network. Inaction or disappearance of those links and nodes alter the structure of the network. We refer such nodes and links as *dull nodes* and *dull links* respectively. Ignoring dull nodes and links may affect analysis and prediction tasks in the network, because these nodes and links provide spurious information. This paper proposes a novel time aware method to predict dull nodes and links at an early stage as a *preprocessing* task in social networks. We formally define the task of predicting such nodes and links at an early stage by exploiting their historical properties: `by observing a time-evolving social network up to time` τ`, predicting the nodes (and links) which will be declared dull at a future time` τ'. The nodes and links, which are idle for long, are labeled as dull. A novel scheme is proposed to fix two time duration thresholds, each for nodes and links of a particular network. If a node or link remains idle longer than its respective threshold, it will be labeled as dull. Removal of predicted dull nodes and links from the network reduces noisy information, and is considered as a *data cleaning* step for dynamic networks.

In recent times, researchers have started exploring temporal dimension in SNA. There are studies [3], [4], [5], [6] on two major SNA tasks namely *link prediction* [7] and *community detection* [8], which consider temporal information as an enhancement to baseline methods. These temporal methods inherently carry properties that can discriminate between an active node (or an active link) and a dull node (or a dull link). However, to the best of our knowledge, none of them has formulated and solved the problem of predicting dull nodes (and dull links), nor considered the time-aware data cleaning for dynamic networks. After pruning the predicted dull nodes and links from the network, the preprocessed network can be used for any SNA task using simple non-temporal baseline methods, which may achieve considerable performance gain along with improvement in running time.

Experiments on two real network datasets demonstrate that the proposed method effectively predicts potential dull nodes and links. We further validate our claim of noise reduction by investigating link prediction [1] performance, subject to data cleaning. We show that, the removal of predicted dull nodes and links results in considerable improvement in performance of state-of-the-art link prediction methods for both datasets.

More specifically, contributions of this paper are as follows.

- It proposes a novel data cleaning method for dynamic networks. This method predicts and removes dull nodes and links, which will eventually become inactive or leave the network in future.
- It also proposes a novel scheme to label the dull nodes and links.
- A case study is presented, which demonstrates the effect of the proposed data cleaning method on state-of-the-art link prediction methods.
- Experiments are performed on two real network datasets, which show that the proposed data cleaning

[1]The link prediction problem in social network is defined as: given the snapshot of a network at time t, predicting the links that will appear at a future time t' [7].

FLIPPER: Fault-Tolerant Distributed Network Management and Control

Subhrendu Chattopadhyay
Department of CSE
IIT Guwahati, India 781039
subhrendu@iitg.ernet.in

Niladri Sett
Department of CSE
IIT Guwahati, India 781039
niladri@iitg.ernet.in

Sukumar Nandi
Department of CSE
IIT Guwahati, India 781039
sukumar@iitg.ernet.in

Sandip Chakraborty
Department of CSE
IIT Kharagpur, India 721302
sandipc@cse.iitkgp.ac.in

***Abstract*—The current developments of software defined networking (SDN) paradigm provide a flexible architecture for network control and management, in the cost of deploying new hardwares by replacing the existing routing infrastructure. Further, the centralized controller architecture of SDN makes the network prone to single point failure and creates performance bottleneck. To avoid these issues and to support network manageability over the existing network infrastructure, we develop Flipper in this paper, that uses only software augmentation to convert existing off-the-shelf routers to network policy design and enforcement points (PDEP). We develop a distributed self-stabilized architecture for dynamic role change of network devices from routers to PDEPs, and make the architecture fault-tolerant. The performance of Flipper has been analyzed from both simulation over synthetic networks, and emulation over real network protocol stacks, and we observe that Flipper is scalable, flexible and fail-safe that can significantly boost up the manageability of existing network infrastructure.**

***Index Terms*—Network architecture, Fault-tolerance, Programmable network, software defined network**

I. Introduction

Consider the following scenario. The network administrator of an academic institute wants to dynamically update the bandwidth distribution policies based on network usage statistics. The institute network is connected with multiple network service providers, and therefore she needs to update the configuration at different edge routers and gateways. With traditional network devices, like layer 3 switches, this task is tedious, as even a minor configuration inconsistency among the edge routers and gateways may lead to severe network underutilization or bandwidth imbalance. Further, the system is also not scalable for such dynamic updates of network configuration policies.

Software defined networking (SDN) [1] is a technique, which can help in dynamic network configuration update. SDN uses centralized controller to convert policies to device configurations, and to update the targeted devices in the network with the corresponding configurations. To enforce policy to configuration conversion, SDN segregates the network control plane from the data plane. The SDN control plane takes care of all the control functionalities and update of device parameters and configurations, whereas the data plane is only responsible for data forwarding based on the configuration parameters set by the control plane.

Although SDN has revolutionized dynamic network management aspects, it requires specific hardwares that can understand the language for SDN, like OpenSwitch [2], [3], so that a SDN controller can dynamically update the configuration parameters for those hardwares. Therefore the important question is: *How much effort and cost do one need to convert an existing network infrastructure to a SDN supported one?* The existing studies in this direction talk about interoperability among SDN supported and non-SDN network devices, such that incremental deployment of SDN supported devices becomes possible [4]–[6]. However, the concern about cost-effectiveness is still there. SDN supported hardwares are much costlier than commercial off-the-shelf (COTS) network devices, and therefore requires huge operational expenditure to replace existing infrastructure by SDN supported infrastructure.

Although it is quite inevitable that the future of network management is SDN, but simultaneously we also ask this question: *Can we make our existing network more management friendly, such that dynamic network configuration becomes possible without much changing the existing infrastructure?* This paper tries to find out the answer of this question. We show that it is quite possible to use the existing COTS routers to work as network policy decision and enforcement points (PDEP), which are known as network information base (NIB). We can turn a COTS router to a NIB by installing a few additional software tools to support "network function virtualization" [7] (NFV). With the help of NFV functionalities, a COTS router can dynamically update the policy control parameters within its neighborhood [8], [9]. Accordingly, we develop a new network management architecture, which is somewhere in-between the traditional architecture and SDN based architecture, where the COTS routers dynamically change their roles from a conventional network router to a NIB, and participate in PDEP functionalities. We call this architecture *Flipper*.

Flipper has two specific advantages over SDN based network architecture, among others. First, to implement Flipper, a network administrator does not need to procure new costly hardwares, and second, Flipper avoids the controller bottleneck problem [4], [7], [10]–[12] which is much debated in the SDN research community. Flipper is a distributed architecture, where the COTS routers execute a distributed self-stabilizing algorithm to decide which nodes can work as a NIB. As

Improving MPTCP Performance by Enabling Sub-Flow Selection over a SDN Supported Network

Subhrendu Chattopadhyay
IIT Guwahati
Guwahati, India 781039
subhrendu@iitg.ernet.in

Samar Shailendra
TCS Research & Innovation
Bangalore, India 560100
s.samar@tcs.com

Sukumar Nandi
IIT Guwahati
Guwahati, India 781039
sukumar@iitg.ernet.in

Sandip Chakraborty
IIT Kharagpur
Kharagpur, India 721302
sandipc@cse.iitkgp.ernet.in

***Abstract*—The primary objective behind the development of Multipath TCP (MPTCP) is to aggregate throughput by creating multiple sub-flows via different network interfaces. A difference in end-to-end path characteristics for the sub-flows may generate out of order segments, causing head of line (HOL) blocking at the receiver. An intelligent selection of a subset of the available sub-flows can reduce the number of out of order segments; thus sub-flow selection can enhance the performance of MPTCP. In this paper, we first propose a Markov model for the performance of MPTCP in terms of end-to-end sub-flow characteristics. Based on the theoretical model, we present an optimization framework for active sub-flow selection by exploiting the controller functionalities over a software defined network (SDN) architecture. Finally, experimental results are obtained to demonstrate performance improvements of MPTCP in terms of aggregated throughput.**

***Keywords*-MPTCP, Sub-flow selection, SDN**

I. INTRODUCTION

Modern day devices are usually equipped with multiple hardware interfaces that can be leveraged to satisfy the demand for increasing traffic by aggregating the available bandwidth at all interfaces. *Multipath Transmission Control Protocol* (MPTCP) [1] has been proposed in the literature as an end-to-end protocol for data-center and enterprise networks with the availability of multi-interface networking devices, which provides the support for bandwidth aggregation via concurrent usage of different interfaces by creating multiple sub-sockets. MPTCP initiates multiple sub-sockets via different interfaces to aggregate the bandwidth.

The current Linux kernel implementation of MPTCP [2] consists of three major modules: *Path Manager*, *Segment Scheduler*, and *Congestion Control Mechanism*. The *Path Manager* module manages the available sub-flows between the end hosts. Currently, MPTCP has proposed two choices of path manager. **(a) Full-mesh path manager** creates sub-sockets for between all available pair of interfaces. On the other hand, **(b) ndiffports** selects k sub-flows among all available sub-flows, where k is a user defined parameter. MPTCP *Congestion Control* module manages congestion window for each sub-flow separately. Several congestion control algorithms like *Linked Increase Algorithm* (LIA) [3], *Opportunistic Linked Increase Algorithm* (OLIA) [4], *Balanced Linked Increase Algorithm* (BALIA) [5] etc. [6], [7] have been proposed for MPTCP. Once the congestion window size for each path is decided, segment scheduler takes the responsibility of scheduling the segments for the individual sub-flows. *Round-Robin* and *lowest RTT First* are the two available segment scheduling strategies described in the MPTCP standard.

The primary task of a segment scheduler is to reduce out of order packets at the receiver. However, in a network, the path characteristics (such as bandwidth, delay, loss rate, jitter etc.) of the underlying sub-flows can be significantly different as well as time varying. The differences in end-to-end path characteristics of each active sub-flow may lead to an increase in out of order segments delivered at the receiver. In [8], the authors have tried to limit the receiver buffer in order to decrease out of order segment delivery by employing network coding. However, it has been found that, their implementation violate MPTCP basic principle of do no harm objective [9]. On the other hand, [10] and [11] focuses on the segment scheduling mechanism to avoid out of order segment generation. However, segment scheduling alone can not reduce out of order delivery and may lead to *Head of Line (HOL) blocking* at the receiver side [12]. HOL blocking increases delays and packet drops. Thus, the number of spurious retransmission also increases. Therefore, Cao *et.al.* [12] proposes a receiver buffer aware path selection mechanism. However, like most of the transport layer protocols, their implementation uses round trip time (RTT) as a measure of path characteristics. In case of MPTCP, one segment and its acknowledgment might follow different paths. So, RTT is not a faithful estimate of a path at the sender side. Therefore, relying on simple RTT driven path characteristics leads to severe performance degradation in MPTCP performance. In our previous work [13] we have shown that MPTCP provides near optimal experience, when the active sub-flows have similar path characteristics, as in those cases RTT provides a good estimation. However, the difference in delay, effective bandwidth, and loss rate can significantly increase the number of out of order segments at the receiver [14]. Therefore, we argue instead of relying on the RTT, MPTCP must rely on end to end path characteristics. Based on the end to end semantics, MPTCP path management module must choose a set of sub-flows which can avoid HOL blocking by reducing out of order delivery [15].

Therefore, in this paper, we propose a *Software Defined Networking* (SDN) [16] aided intelligent dynamic path management scheme. SDN provides a logically centralized view of network topology parameters to the application protocols

PTC: Pick-Test-Choose to Place Containerized Micro-services in IoT

Shubha Brata Nath*, Subhrendu Chattopadhyay†, Raja Karmakar‡,
Sourav Kanti Addya§, Sandip Chakraborty¶ and Soumya K Ghosh‖
*§¶‖Indian Institute of Technology Kharagpur, India † Indian Institute of Technology Guwahati, India
‡Techno International New Town, India
Email: *nath.shubha@gmail.com, †subhrendu@iitg.ac.in, ‡rkarmakar.tict@gmail.com,
§kanti.sourav@gmail.com, ¶sandipc@cse.iitkgp.ac.in, ‖skg@cse.iitkgp.ac.in

***Abstract*—In the presence of the Internet of Things (IoT) devices, the end-users require a response within a short amount of time which the cloud computing alone cannot provide. Fog computing plays an important role in the presence of IoT devices in order to meet such delay requirements. Though beneficial in these latency-sensitive scenarios, the fog has several implementation challenges. In order to solve the problem of micro-service placement in the fog devices, we propose a framework with the objective of achieving low response time. This problem has been formulated as an optimization problem to improve the response time by considering the time-varying resource availability of the fog devices as constraints. We propose an orchestration framework named *Pick-Test-Choose* (PTC) to solve the problem. PTC uses *Bayesian Optimization based iterative reinforcement learning* algorithm to find out a micro-service allocation based on the current workload of the fog devices. PTC employs containers for service isolation and migration of the micro-services. The proposed architecture is implemented over an in-house testbed as well as in iFogSim simulator. The experimental results show that the proposed framework performs better in terms of response time compared to various other baselines.**

***Keywords*-Fog Computing; Internet of Things; Micro-service Placement; Container Placement; Reinforcement Learning**

I. Introduction

Usage of cloud infrastructure is a popular choice for the deployment of large scale distributed applications. However, the cloud is not a suitable platform for many Internet of Things (IoT) applications. Short bursty flows generated by the IoT applications increase the response time when each of the packets is processed in the cloud. To overcome this limitation, Cisco proposed *"Fog computing"* [1], which employs *"in-network processing"* to deliver user desired quality of service (QoS) for time critical IoT applications. Fog hosts small-scale cloud to support *"in-network processing"* by using the residual resources of the network equipment like the routers. Fog devices are resource constrained; therefore, instead of using monolithic *"service oriented architecture"* (SOA), the fog applications or services must adopt *"micro-service architecture"* [2], where each application is developed in the form of a bunch of loosely coupled lightweight services. Although the placement of micro-services over various fog devices is as important as the service placement in case of cloud computing, it is not very straightforward and differs from the traditional service offloading mechanisms [3], [4]. The primary challenges are as follows. (i) As the fog uses in-network processing, all the fog devices have their primary workloads (e.g., routing for a network router), and the fog resource availability varies over time. Further, the secondary workload from the micro-services should not interfere with the primary workload of the fog devices. (ii) Multiple micro-services can be placed simultaneously at the fog devices; therefore, the architecture requires micro-service isolation to ensure resource provisioning. (iii) Based on the temporal nature of the primary workload, the architecture must support seamless migration of micro-services from one fog device to another to ensure application QoS. (iv) The deployment framework and the micro-service migration framework must be lightweight so that it can cater to low latency IoT applications by deciding micro-service placement quickly. (v) The highly dynamic nature of the in-network processing architecture increases monitoring overhead, and maintaining consistency of the monitored statistics is difficult. The monitoring inconsistency leads to the performance degradation of the overall system. Ensuring these five requirements simultaneously is non-trivial over a fog environment.

A few works in the recent literature have addressed the problem of micro-service placement in the fog devices. In [5], the authors have addressed the application placement problem by placing the virtual machines (VM) in the edge or fog devices. However, the use of VM requires high resource consumption; to avoid this overhead, [6] have proposed a container (lightweight virtualization technology) driven framework to speed-up application deployment procedure. [7] and [8] have proposed the container placement and module mapping algorithm respectively in a hierarchical fog environment. DROPLET [9] provides a computation partitioning and offloading procedure. On the other hand, a micro-service offloading framework by exploiting traditional memory allocation strategies have been proposed in [10]. In Foglets [11], the authors have proposed a greedy service placement strategy for a hierarchical fog infrastructure in the presence of mobile users to minimize the access delay. For this type of mobility capable fog systems, Mobility-based [12] have provided a VM placement and migration framework to maximize the number of applications placed in fog while reducing the overall application latency. However, most of these existing works