Software Defined Disruption Tolerant Networks

A thesis submitted

in partial fulfillment for the award of the degree of

Doctor of Philosophy

by

Sarath Babu

Department of Avionics

Indian Institute of Space Science and Technology

Thiruvananthapuram, India

May 2021

Certificate

This is to certify that the thesis titled ***Software Defined Disruption Tolerant Networks*** submitted by **Sarath Babu**, to the Indian Institute of Space Science and Technology, Thiruvananthapuram, in partial fulfillment for the award of the degree of **Doctor of Philosophy** is a bona fide record of the original work carried out by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. B. S. Manoj
(PhD Advisor)
Professor
Department of Avionics
IIST

Dr. Deepak Mishra
Professor & Head
Department of Avionics
IIST

Place: Thiruvananthapuram
Date: May 2021

Declaration

I declare that this thesis titled ***Software Defined Disruption Tolerant Networks*** submitted in partial fulfillment for the award of the degree of **Doctor of Philosophy** is a record of the original work carried out by me under the supervision of **Dr. B. S. Manoj**, and has not formed the basis for the award of any degree, diploma, associateship, fellowship, or other titles in this or any other Institution or University of higher learning. In keeping with the ethical practice in reporting scientific information, due acknowledgments have been made wherever the findings of others have been cited.

Place: Thiruvananthapuram
Date: May 2021

Sarath Babu
(SC14D004)

This thesis is dedicated to

Dorothy Rowe
(1930–2019)

"*Yet we cannot look at the world with eyes washed clean of all our past experience. All we can do is, first, to acknowledge that this is how we see, and then set about practicing how to create alternative interpretations. An alternative interpretation might prove to be closer to the truth than the first interpretation.*"— Dorothy Rowe, Author and Psychologist

Acknowledgements

As Jeff Mach wrote "*I am a product of the flames which burnt me; the anvil which forged me; and the will that made me grow formidable instead of breaking.*" I am so blessed to have had an environment during the period of my PhD work that provided me such a will to walk through my journey of life with a new perspective and attitude.

First, I express by sincere gratitude to my PhD advisor *Dr. B. S. Manoj*, Professor, Department of Avionics, Indian Institute of Space Science and Technology (IIST), for taking me as one of his PhD students and opening up my mind toward the world of scientific research. His immense pragmatic experience in industry and academia, as well as the very patient approach in exploring and resolving problems helped me to realize my PhD work a life-experience rather than an activity for graduation. His involvement as well as non-involvement provided me sufficient freedom in all activities during the PhD work. I thank him specifically for loading (not overloading) me with research and teaching assistantship work so that I hardly get any time to think on anything other than academics. More importantly, his readiness to listen to all problems, both academic as well as personal, *at anytime* makes him a precious gem in my heart.

I am so grateful to the members of my Doctoral Committee, *Dr. N. Selvaganesan*, Professor, Department of Avionics, IIST, *Dr. Venkata Ramana Badarla*, Associate Professor & Head, Department of Computer Science & Engineering, Indian Institute of Technology, Tirupati, *Dr. Priya Chandran*, Professor, Department of Computer Science & Engineering, National Institute of Technology, Calicut, *Dr. Elizabeth Sherly*, Director & Professor, Indian Institute of Information Technology & Management, Kerala, *Dr. Rajeevan P. P.*, Associate Professor, Department of Avionics, IIST, *Dr. Anil Kumar C. V.*, Professor, Department of Mathematics, IIST, and *Dr. Sumitra S.*, Associate Professor, Department of Mathematics, IIST, for reviewing my work and providing timely feedback and suggestions throughout the duration of the PhD work. I extend my gratitude to *Dr. Vineeth B. S.*, Assistant Professor, Department of Avionics, for providing suggestions on the simulator developed as part of the research and reviewing the documentation. I value *Dr. Chris Prema S.*, Assistant Professor, Department of Avionics, IIST, for being a dear friend during the course of my PhD studies.

It was an eye-opening experience to work with *Dr. Nalini Venkatasubramanian*, Professor, University of California, Irvine, during the initial days of my research work as part

of an Indo-US collaborative project. Weekly discussions with *Prof. Nalini* and her students *Mr. Kyle E. Benson* and *Mr. Ranga Raj* helped me a lot in developing a new way of thinking toward research and, further, designing my research direction.

It was an immense pleasure to work with *Dr. Abhishek Chakraborty*, my senior research scholar and currently Senior Project Officer at the Department of Computer Science & Engineering, Indian Institute of Technology, Madras, who provided me insights on his research field which I applied in different threads of my research domain. I cannot express in words the companionship provided by *Mr. Debabrata Dalai* and *Mr. Avinash Chalumuri*, research scholars of our Systems and Networks Lab (SNL), IIST, which made my life at the lab more enjoyable. I extend my respect and gratitude to *Dr. Dharmendra Singh Yadav* and *Dr. Prescilla Koshy*, former post-doctoral fellows of SNL, for providing me insights on different domains of next generation communication systems.

My research-related experiments would have been difficult without the presence of *Mrs. Divya R. S.*, lab staff of SNL, who played the most important role in indenting, purchasing, and making the hardware components available on time for the experiments. In addition, she helped me in executing simulations in our lab and sending me the related files during the COVID-19 pandemic. Besides *Mrs. Divya*, other technical and lab staffs *Mrs. Ananthalakshmi S.*, *Mrs. Preeti Yadav A. G.*, *Mrs. Dhanya G. D. Nair*, *Mrs. Archana K. P.*, *Mrs. Smitha S.*, *Mrs. Lekshmi Devan A.*, *Mr. Abhilash S.*, and *Mr. Nidheesh Ravi*, for providing me a homely atmosphere in our department. At times, they acted as mobile wireless agents for my mobility-related research experiments. I also thank *Mrs. Preetha Sajeev*, office staff, Department of Avionics, IIST, for instantaneously providing all help whenever required.

Life at IIST would have been nothing for me without the undergraduate and postgraduate students of IIST, as well as the students from other institutions who had done internship in our lab during the period of my PhD work. It was a great experience to share knowledge with them as part of my teaching assistantship and, more importantly, learn many new things from them. Discussions during their internships and projects helped me in exploring new areas, thereby, enriching my knowledge in the communication domain. I appreciate their friendliness and, at times, hour-long phone calls.

Beyond words are the relationships that I got from my PhD colleagues at IIST, especially *Mr. Ameya Anil Kesarkar, Mrs. Richa Sharma, Mr. Sabu M., Mr. Najeeb P. K., Mr. Rakesh R., Mr. Sujith Vijayan, Mr. Swagat Ranjan Das, Mrs. Jayalekshmi N. S., Mr. Dibyendu Adak, Mr. Randeep N. C., Mr. Deepak M., Mr. Bibin Johnson, Mr. Rahul. O. R., Mr. Deepak Gopalakrishnan, Mr. Richu Sebastian C., Mrs. Anuja Vijayan, Mr. Rajkumar R., Mr. Vijayakumar*

Muni, Mr. Nikhilraj A., Mr. Sathishkumar P., Mr. Praveen Kumar Kodakkal, Mr. Rajesh N., Mr. Mahesh T. V., Mr. Prabith K., Mr. Muhammed Sihas K. M., Mr. Dubacharla Gyaneshwar, Mr. Vinod Kumar P., Mr. Gourahari Nayak, Mr. Elangovan K., Mr. Sarath K. P., Mr. Sonu Bose, Mr. Sajith V. S., Mr. Renjith Thomas, Mr. Pramod Martha, Mr. Jeeva B., Mr. Sriram S. Kumar, Mrs. Gopika R., Mrs. Elizabeth George, Mrs. Anjitha R. G., Mr. Pavanam Thomas, Mr. Kirubakaran C., Mr. Nibin Raj, Mr. Sreehari B. Nair, and *Mr. Binoy Babu*. It is an enjoyable time at the hostel as well as at the mess discussing non-academic matters with them. I also value my friendship with *Ms. Rashi Jain*, currently PhD scholar at University of Strasbourg, for her nice conversations and discussions.

I thank the IEEE Student Branch, IIST, for providing me a platform for conducting workshops on Python programming language and LaTeX, as well as for organizing and volunteering several technical events which helped me to improve my skills in organizing different events. I extend my gratitude to *Dr. Sam K. Zachariah*, Adjunct Professor, Department of Avionics, IIST, for provoking me to dig deep into the LaTeX documentation system by pointing out errors while designing official report templates.

I value the encouragement and support provided by my teachers, *Shri. Dhandapani K. R.* and *Shri. Rajesh A. V.*, for all my academic endeavors since my higher secondary school days. I appreciate the friendship and hospitality offered by my family friends *Shri. K. N. Chithambaran* and *Smt. T. N. Latha* during my pre-synopsis meeting and the thesis writing period. Finally, I thank my parents *Shri. Gopalakrishnan P. S.* and *Smt. Visala T. K.*, and my brother *Shri. Hemanth Babu* for *'disrupting'* my research mood with their phone calls.

Sarath Babu

Abstract

Wireless networks play a crucial role in our activities of daily living, especially due to the surge of mobile devices since 1990. Apart from the widely-accepted domains of cellular and local area networking, wireless networks find their applications in tactical networks, emergency response systems, underwater networks, satellite networks, and emerging Internet of Things (IoT). However, the limited communication range, bandwidth constraints, as well as the mobility of end-user devices make the seamless end-to-end communication a challenging task in such networks. In addition, the traditional networking frameworks seem inadequate to cater the dynamic application-specific requirements of users and business community. Software Defined Networking (SDN) emerged as a new networking paradigm in this regard to provide flexibility in network control with the separation of control plane from data plane which are otherwise coupled together in traditional networking devices. In SDN, the devices in data plane, known as *switches*, are considered as non-intelligent devices which forward the packets as per the rules dictated by a logically centralized control plane point known as the *controller*. The data and control planes are assumed to have a secure and reliable communication channel to exchange the network control decisions. However, such an assumption becomes invalid in wireless environments that involve node mobility and frequent link disruptions.

In this thesis, we propose an SDN framework for wireless environments that involve link disruptions and mobility of nodes, *including switches and controllers*. As the first step toward adopting SDN to such disruption-prone wireless networks, we design a novel resource discovery and self-configuration scheme to enable the SDN devices to identify and adjust themselves depending on the changing network conditions. In order to capture the network dynamics as well as to identify the SDN resources in a network, we extend the existing Optimized Link State Routing (OLSR) protocol to Software Defined OLSR (SD-OLSR) and provide it as the medium for switch-controller communication. In addition, two controller handoff schemes, Switch-Initiated Handoff (SIH) and Controller-Initiated Handoff (CIH), are designed to achieve the self-configuration of switches and controllers. The results from an experimental wireless mesh network testbed show that CIH is appropriate for environments offering easy on-the-fly configuration of nodes while SIH is best suited for real-world time-sensitive and strategic networks.

In order to cope with the link disruptions between devices in data plane, we proposed a novel Software Defined Disruption Tolerant Networking (SD-DTN) framework with controlled buffering of packets at the switches using a proposed STORE action. In SD-DTN, the controller exploits switches' memory to buffer packets during link disruptions and forward them toward the destination when appropriate links get established. We model the network involving link disruptions using temporal graphs and Markov chain, and propose an SDN controller algorithm Earliest Arrival Path with Minimal Storage Time (EAPMST) to compute the buffering/routing decision for the switches. Our SD-DTN is realized using a new SDN switch architecture integrated with a storage subsystem to handle the controlled buffering operation. The performance of our framework is tested using a disruption-prone wireless mesh network testbed deployed in the campus building and the results show an improvement in throughput beyond 25% with random mobility model.

Finally, we explored two emerging application domains of SD-DTN: (*i*) Software Defined Vehicular Networks (SD-VNs) and (*ii*) Software Defined Satellite Networks (SD-SNs). In order to improve the vehicles' contacts with the wireless points deployed at the roadsides, we define a new metric *Effort* for road networks and devise an *Effort*-based Roadside Unit (RSU) placement scheme for SD-VNs considering the structural properties of road networks. On the other hand, the performance of EAPMST algorithm is studied in the context of SD-SNs using two satellite constellation models, *Constrained*-Iridium-NEXT and DebriNet. Further, we discuss the scope and importance of our self-configuration scheme and SD-DTN framework in software defined tactical and emergency response networks. Our studies reveal that the SD-DTN framework along with the EAPMST algorithm is capable of buffering and carrying packets, and maintaining the communication sessions during disruptions, thereby, making our approach a suitable candidate for efficient network control in next generation disruption tolerant wireless networks.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations

5G	5th Generation
6G	6th Generation
AP	Access Point
API	Application Programming Interface
BC	Betweenness Centrality
BS	Base Station
CC	Command and Control Center
CIH	Controller-Initiated Handoff
COTS	Commercial Off-The-Shelf
CPP	Controller Placement Problem
DTN	Delay/Disruption Tolerant Network
EAPMST	Earliest Arrival Path with Minimal Storage Time
EBC	*Effort*-based Betweenness Centrality
EoC	Earliest-on-Contact
ETX	Expected Transmission Count
FTP	File Transfer Protocol
GEO	Geosynchronous Equatorial Orbit
HM	Handoff Module
HTTP	Hypertext Transfer Protocol
IAN	Incident Area Network
ICMP	Internet Control Message Protocol
IIST	Indian Institute of Space Science and Technology
IoT	Internet of Things
IoV	Internet of Vehicles
IP	Internet Protocol
ISL	Inter-Satellite Link
ISRO	Indian Space Research Organisation

LAN	Local Area Network
LBC	Length-based Betweenness Centrality
LEO	Low Earth Orbit
LoRa	Long Range
LoS	Line-of-Sight
MAC	Medium Access Control
MANET	Mobile Ad hoc Network
MDC	Mobile Data Collecting Agent
MPR	Multi-Point Relay
NFV	Network Function Virtualization
NH	No Handoff
NH-CF	No Handoff with Controller Failure
NIC	Network Interface Card
OBC	On-Board Computer
OLSR	Optimized Link State Routing
PDF	Probability Density Function
PS4	PSLV Stage-4
PSLV	Polar Satellite Launch Vehicle
QoS	Quality of Service
RNN	Recurrent Neural Network
RSU	Roadside Unit
RTT	Round-Trip Time
SD-DTN	Software Defined Disruption Tolerant Network
SDN	Software Defined Network
SD-OLSR	Software Defined Optimized Link State Routing
SD-SN	Software Defined Satellite Network
SD-VN	Software Defined Vehicular Network
SD-WLAN	Software Defined Wireless Local Area Network
SD-WMN	Software Defined Wireless Mesh Network
SD-WN	Software Defined Wireless Network
SIH	Switch-Initiated Handoff
STK	Systems Tool Kit
SUMO	Simulation of Urban MObility
TCAM	Ternary Content Addressable Memory
TCP	Transmission Control Protocol

UAV	Unmanned Aerial Vehicle
UDP	User Datagram Protocol
UDTNSim	Urban Delay Tolerant Network Simulator
UE	User Equipment
V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle
V2X	Vehicle-to-Everything
VoIP	Voice over Internet Protocol
WLAN	Wireless Local Area Network
WMN	Wireless Mesh Network
WSN	Wireless Sensor Network

Nomenclature

$\{\}$	Empty set
$\langle\,\rangle$	Empty graph sequence
$\langle\langle\,\rangle\rangle$	Empty temporal path
$\mathbb{C}$	Set of SDN controllers
$\mathbb{S}$	Set of SDN switches
α	Smoothing factor
τ	Handoff threshold
$\delta^t_{u,v}$	The cost, derived from ETX, between nodes u and v at time t
$\mathcal{X}^t_{u,v}$	ETX between nodes u and v at time t
$\mathcal{S}$	Sequence of static graphs
$\mathcal{S}'_t$	Predicted graph sequence at time t
$G_t = (V_t, E_t)$	Graph at time t with vertex set V_t and edge set E_t
G'_t	Predicted graph at time t
$\mathbb{N}$	Set of natural numbers
$X_{(u,v),t}$	State of edge (u, v) at time t
$\varnothing$	Null value or the absence of a set/path
$P_{(u,v)}$	Probability transition matrix of link (u, v)
$Q_{(u,v),t}$	Probability distribution of $X_{(u,v),t}$ at time t
M	Number of predicted future time-steps
$\mathcal{P}$	Temporal path
h_v	Minimum distance from source node to node v
p_v	Predecessor of node v in the earliest arrival path from source node
c_v	Cumulative buffering time till node v in the earliest arrival path
$\mathcal{N}^G_v$	Neighbors of node v in graph G
$\mathbb{V}$	Set of visited nodes
ξ	Buffering threshold
$C_B(v)$	Betweenness centrality of node v

$\mathcal{F}_{u,v}$	*Effort* for travelling from node u to v through the link (u, v)
$\mathbb{R}^+$	Set of positive real numbers
$l_{u,v}$	Length of the link (u, v)
$t_{u,v}$	Type of the link (u, v)
$G^{\mathcal{F}}$	*Effort*-graph
N	Number of RSUs
μ	Inhibition distance
$\mathcal{L}$	List of nodes arranged in the descending order of their BCs
$\mathcal{R}$	Set of RSU locations
G_p	Preference graph

Chapter 1
Introduction

"*This is the age of disruption.*"
— Sebastian Thrun

"*All disruption starts with introspection.*"
— Jay Samit

Wireless communications had a revolutionizing role in information dissemination among people, especially when they are on the move. Due to the massive transformation since early 1900s, wireless mode of communication infiltrated into every aspect of human life from radio broadcasting to the recent Internet of Things (IoT) assisted daily living. Even though the technological advancements catalyze societal progress, the state-of-the-art does not seem sufficient to satisfy the nature of human exceptionalism. As pointed out in [1], the "*trends indicate that people want to communicate while on the move. They also want to take their computers wherever they go. It is therefore reasonable to assume that people want the same data communication capabilities on the move as they enjoy in their home or office.*" Further, authors of [1] suggested that "*Many technical challenges must be overcome to improve wireless network performance such that users will accept this performance in exchange for mobility*" in order to achieve the wireless vision that allows people to "*operate a virtual office anywhere in the world using a small handheld device — with seamless telephone, modem, fax, and computer communications.*" Following the wireless vision, next generation communication networks, especially 5^{th} Generation (5G) [2, 3] and 6^{th} Generation (6G) [4, 5], are envisioned to offer very high data-rate (in the order of gigabits per second) and extremely low round-trip latency (below 1 millisecond) irrespective of the location and mobility of end-users.

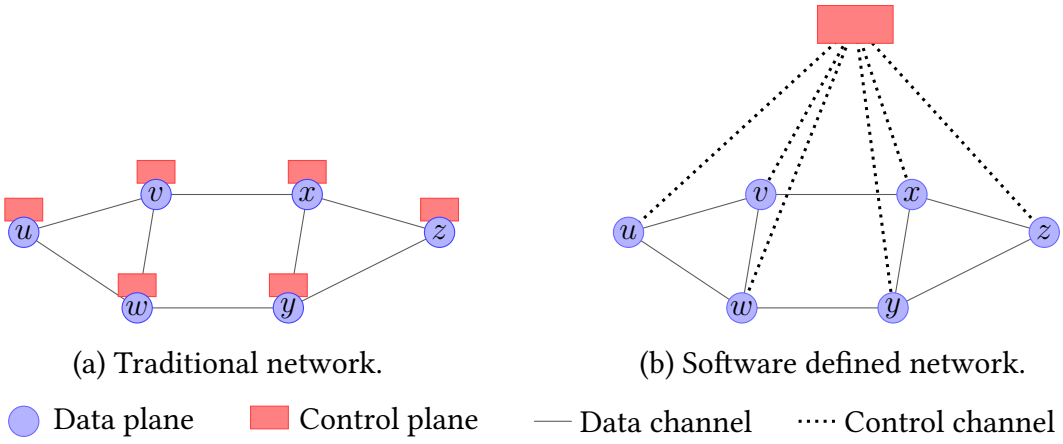
Major challenges in achieving the above-mentioned wireless vision stem from the physical mobility of wireless (end-user) devices. One among them is the link disruptions

— the major focus of the thesis — emanating from the fluctuations in wireless medium as well as the limitations, especially the communication range of wireless transceivers. Moreover, the dynamic user/application specific requirements add further complexity to the entire network system design. Therefore, it is inevitable for wireless communication systems to have a flexible design that "*... must be optimized for this channel and must be robust and adaptive to its variations as well as to network dynamics. Thus, these networks require integrated and adaptive protocols at all layers, from the link layer to the application layer*" [6]. Software Defined Networking (SDN) [7] has the innate potential for achieving such a flexible network design, a prerequisite for next generation wireless networks.

1.1 Software Defined Networks: Preliminaries

In traditional networks as shown in Figure 1.1(a), the control plane which computes the forwarding decisions on the packets entering a node and the data plane which forwards the packets according to the control decisions from control plane reside within the networking device. Such a strong coupling between the control and data planes prevents network administrators from realizing a flexible network control. Moreover, the high dependence on hardware vendors and the proprietary nature of control algorithms hamper innovation in the networking domain. Therefore, substantial efforts have been made to separate control plane from the networking device, thereby, making the network control an easy process. Even though the idea of such a separation finds its roots in [8–10], software defined networking is considered as the first pragmatic approach toward managing networks from a logically centralized control point. SDN decouples the control plane from traditional networking devices and makes it available to the network programmers in order to manage the network elements depending on the changing user requirements and network dynamics. Figure 1.1(b) depicts the separation of control plane in SDN with respect to the traditional networking approach shown in Figure 1.1(a).

In SDN, the traditional networking devices such as routers and switches are replaced with generic data forwarding devices known as *SDN switches*. Switches forward the data packets as per the network control rules, known as *flow-rules*, dictated from the control plane device, known as *SDN controller*. A network *flow* is defined as a sequence of packets sharing specific attributes of interest. For example, all data packets destined to node v in Figure 1.1 can be considered as a flow if there exists a requirement to control the traffic toward node v. Controller and switches communicate through a control channel (depicted as dotted lines in Figure 1.1(b)) using a south-bound interface, the most popular being the



(a) Traditional network. (b) Software defined network.

Figure 1.1: Traditional networking vs. SDN.

OpenFlow protocol [11]. The control channel can be a separate dedicated network, known as *out-of-band* control, or can share the same network infrastructure of the data plane, known as *in-band* control. The controller is assumed to have the global network view which is used to compute the flow-rules. The flow-rules are further sent to the switches for handling packets corresponding to each flow. Each switch is integrated with *flow-tables* to store the required flow-rules. Figure 1.2 shows an example flow-table with three flow-rules.

Match-fields			Actions	Flow-rule attributes		
IP Src.	**IP Dst.**	**· · ·**	**Action**	**Hard Timeout**	**Idle Timeout**	**· · ·**
10.10.10.20	*	*	PORT 1	120	60	–
*	10.10.10.50	*	DROP	–	60	–
*	*	*	CONTROLLER	–	–	–

Figure 1.2: Example of an SDN switch's flow-table.

A flow-rule within a flow-table is composed of three parts: (*i*) match-fields, (*ii*) actions, and (*iii*) flow-rule attributes.

1. **Match-fields:** The attributes of a packet corresponding to each layer, specifically Layers 2–4 of the TCP/IP protocol stack, along with the ingress port constitute the match-fields. Examples of match-fields include source and destination IP addresses (as shown in Figure 1.2), source and destination Medium Access Control (MAC) addresses, and the source and destination ports of the transport layer protocols. OpenFlow v1.0 [12] defines 12 match-fields while OpenFlow v1.5 [13] improves the granularity to 45 attributes.

2. **Actions:** The actions that need to be taken on the packets which have matched with a flow-rule are specified in the *Action* field. Examples of actions include OUTPUT (to the specified port), DROP, CONTROLLER, as well as the modification of packet attributes. In this thesis, we introduce a new action STORE for the controlled buffering of packets in SDN switches, thereby, handling link disruptions in software defined wireless networks.

3. **Flow-rule attributes:** Different counters associated with a flow-rule constitute the flow-rule attributes. Examples include idle-timeout, hard-timeout, rule priority, and the number of rule-hits. Idle-timeout defines the lifetime of a flow-rule without a rule-hit, i.e., the rule gets removed from the flow-table if no packets matches the rule within the time unit specified as idle-timeout. On the other hand, hard-timeout removes the flow-rule after the specified time units irrespective of a flow-rule hit or miss. The flow-rule attributes are sent to the controller on request for generating the global network view.

Upon the arrival of a packet, the switch compares the packet's attributes against the match-fields of the flow-rules for a match (or alternatively called rule-hit). In case of a rule-hit, the action specified in the flow-rule's *Action* field is applied on the packet. For example, in Figure 1.2, the packets from the source address $10.10.10.20$ are forwarded to Port 1 of the switch while the packets destined to address $10.10.10.50$ are dropped. The match-fields with wildcard entries are discarded from matching with the packet attributes. The last rule in Figure 1.2, with all match-fields wildcarded, represents a flow-rule miss, i.e., the packet does not have a match with any of the specified flow-rules. In case of such a flow-rule miss, the packet is encapsulated in a flow-rule request message, known as PacketIn, and sent to the controller. The controller disassembles and analyzes the packet, computes the flow-rule, and sends the rule to the requested switch using a FlowMod message. The rule gets stored in the switch's flow-table for handling subsequent packets of that flow. It is important here to note that a controller can also send flow-rules to any switch, without a PacketIn message, on a proactive basis as well. In short, using the flow-rules, a controller can acquire the control of each switch (and, thereby, the entire network) for handling the network flows. SDN simplifies such a network management by handing over the network control to administrators so that they can program the network depending on the end-user requirements and network dynamics.

1.2 SDN in Wireless Environments

The concept of SDN was originally developed for wired environments such as campus and enterprise networks [11]. However, the surge of mobile devices makes the concept suitable for wireless networks due to their dynamism resulting from the uncertainty of wireless medium as well as user mobility. Compared to wired infrastructures, wireless environments face several challenges including scarcity of wireless medium, interference in communication channel, limited communication range of devices, and finally, the random mobility pattern of devices. Such limitations make wireless networks prone to frequent *link disruptions*, thereby, degrading the network performance. In this thesis, we analyze the influence of link disruptions in Software Defined Wireless Networks (SD-WNs) and design solutions for tolerating such disruptions in order to achieve the required network performance.

In SD-WNs, the link disruptions occur primarily at two levels: (*i*) in control channel, i.e., between data plane and control plane, and (*ii*) in data plane, i.e., between devices in data plane.

1.2.1 Link Disruptions in SDN Control Channel

Besides data and control planes, a reliable and secure control channel forms the backbone of any SDN infrastructure. That is, control channel forms the communication platform for network control decisions to reach SDN switches in order to achieve the desired network control. Even though a secure and reliable control channel was envisioned in [11], the presence of link disruptions in SD-WNs makes its realization a difficult task. In practice, the control channel operates either in out-of-band mode or in in-band mode as mentioned in Section 1.1.

Figure 1.3 shows an example network with out-of-band control where the data and control packets are communicated through separate channels. As shown in Figure 1.3(a), all switches reside within the communication range of the controller at time $t = 0$. At $t = 1$, two switches y and z move out of the controller's footprint and become outside the purview of network control (as in Figure 1.3(b)), thereby, compromising the objectives of SDN.

Unlike out-of-band control, in in-band control, the same communication channel is shared between data and control packets as shown in Figure 1.4. Similar to the case shown in Figure 1.3, switches y and z move out of the communication range of the controller in

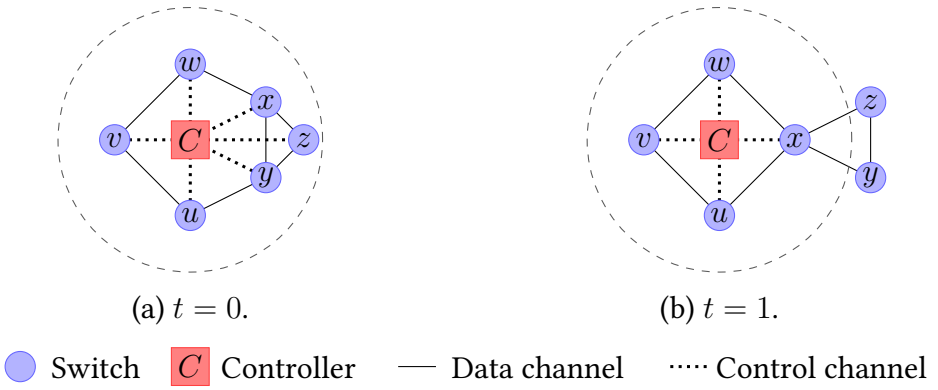


Figure 1.3: Control channel operating in out-of-band mode. The dashed circle depicts the communication range of the controller.

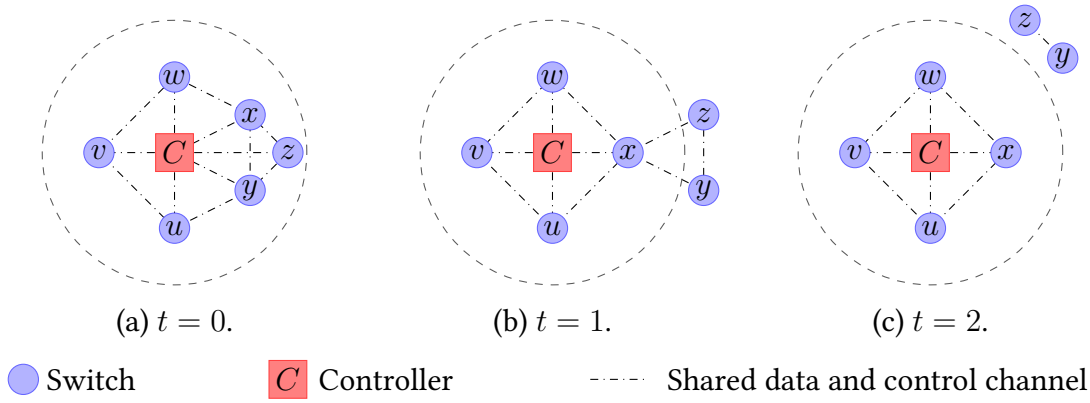


Figure 1.4: Control channel operating in in-band mode. The dashed circle depicts the communication range of the controller.

Figures 1.4(b) and 1.4(c). However, at time $t = 1$, nodes y and z can reach the controller via node x due to the shared communication channel. In Figure 1.4(c) the network gets partitioned into two and the nodes y and z do not have a path toward controller C. That is, in in-band control, link disruptions in the shared channel form a major bottleneck in communicating the network control decisions from controller to the switches. Moreover, due to the shared channel, control packets are also subject to dynamic network behaviors such as congestion and user mobility. Therefore, in-band control is more suitable for networks involving mobile nodes, such as Wireless Mesh Networks (WMNs), Delay/Disruption Tolerant Networks (DTNs), and Wireless Sensor Networks (WSNs), where out-of-band control is infeasible due to node mobility.

1.2.2 Link Disruptions in SDN Data Channel

In any wireless network, the link disruptions occur primarily due to the mobility of end-user devices. Network control incurs less effort in mobile environments with predictable node mobility patterns such as satellite networks and public-transport-based vehicular networks. On the other hand, network control becomes a challenging task under uncertain mobility pattern of nodes as the case with tactical networks and emergency response networks, which also require real-time network control decisions. Link disruptions in data channel result in Disruption Tolerant Networking (DTN) scenarios [14] and are traditionally handled using a *store-carry-and-forward* approach for the packets by introducing a bundle layer [15] in the TCP/IP protocol stack. The bundle layer, running bundle protocol [16], in each node stores the packets in the absence of appropriate links and are forwarded when the links get established.

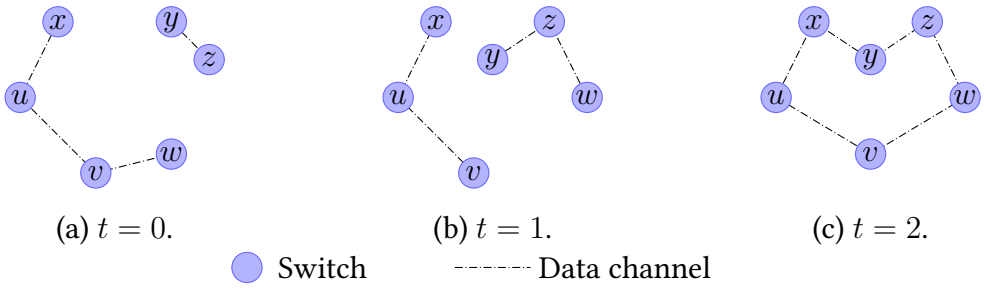


Figure 1.5: Example disruption tolerant networking scenario in data plane.

Consider an example DTN scenario shown in Figure 1.5, where node u wants to send data to node w and each link costs 1 time unit to carry the data. The packets can be routed via node v since there exists a path $u \rightarrow v \rightarrow w$ at time $t = 0$. However, at time $t = 1$, the link (v, w) gets disrupted and the packets residing at node v are buffered until v comes in contact with either w or any other node which can subsequently deliver the packets to node w as at time $t = 2$. At $t = 1$, node u can forward the packets to either node v or x. However, the forwarding decision depends on several factors such as the mobility pattern of nodes, the probability of a path toward node w from nodes v and x, and the buffer space available at the nodes. Even though different algorithms are designed for DTNs to improve end-to-end delay and throughput, such algorithms may not be suitable for the dynamics of future communication systems characterized by changing user/business requirements and high degree of mobility. Therefore, we propose an SDN-based solution with controlled buffering for DTNs to meet the requirements of disruption-prone next generation wireless networks.

1.3 Motivation behind Software Defined DTNs

The primary motivation behind the research toward Software Defined Disruption Tolerant Networks (SD-DTNs) stems from our analysis on information gathering in a shanty town emergency response system [17, 18]. Shanty towns are characterized by complex road networks, poor infrastructure, and high degree of underdevelopment. Therefore, disasters have detrimental impacts on shanty towns compared to well planned cities. It is important in such scenarios to gather maximum information from the disaster location within minimum time in order to design efficient emergency response systems.

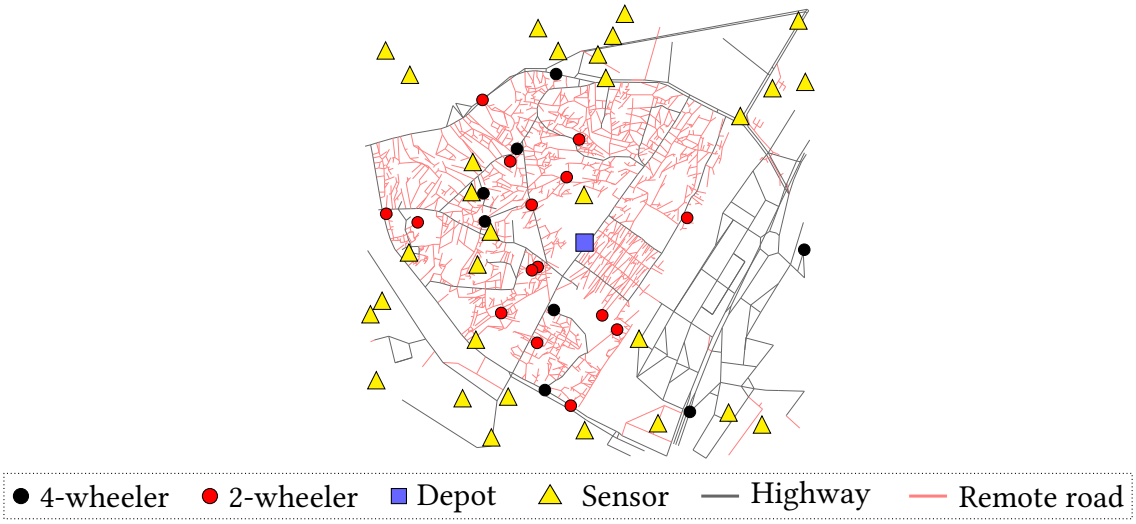


Figure 1.6: Snapshot of a multi-agent-based post-disaster information gathering scenario from shanty town of Dharavi, Mumbai, India.

Figure 1.6 shows an emergency response scenario from shanty town Dharavi in India. Sensors (depicted as Yellow triangles) are assumed to be dropped from helicopters or Unmanned Aerial Vehicles (UAVs) to the area at random locations to sense different environmental parameters. We make use of Mobile Data Collecting agents (MDCs) [17–19], i.e., vehicles such as 2-wheelers and 4-wheelers equipped with wireless interfaces, to locate the sensors, gather data from them, and communicate the data to the depot (depicted as Blue-filled square in Figure 1.6) where the emergency response decisions are taken. Due to the complex road network characteristics, MDCs have restrictions on accessing different types of roads. For example, the road network in Figure 1.6 consists of two types of roads, highways (depicted as Black lines) and remote roads (depicted as Red lines). Remote roads are attributed to their narrow width and are accessible only to 2-wheeler MDCs while highways are wide roads capable of accommodating both 4-wheeler and

2-wheeler MDCs. Due to the lack of communication infrastructure and limited communication range, MDCs do not have direct links with the depot which results in a DTN scenario. Therefore, the 2-wheeler MDCs gather data from the sensors located at remote locations, store the data, and handover the data to the other 2/4-wheeler MDCs when they come in contact with each other, and subsequently deliver the data to the depot. It is important here to take the store/forward decisions on a real-time situational basis unlike deterministic network control algorithms in traditional DTNs.

1.3.1 Disruption-prone Wireless LAN Environments

Wireless Local Area Networks (WLANs) form the most popular form of wireless data communication infrastructure among home, campus, and enterprise environments. WLANs make use of Access Points (APs) which enable the end-user devices to connect to the network using wireless medium, thereby, providing access to the Internet on the move. However, the APs are usually connected to the gateway using a wired channel.

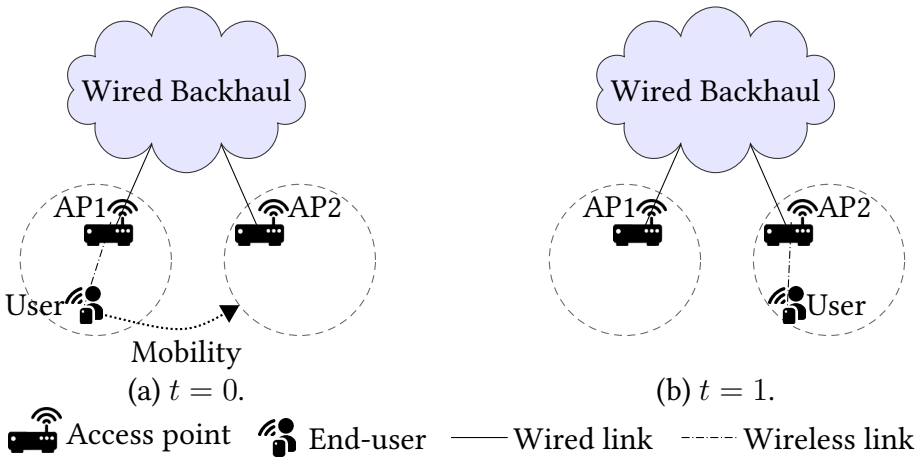


Figure 1.7: Wireless LAN environment involving link disruptions. The dashed circles depict the communication range of APs.

Consider an example WLAN scenario with two APs and a single user shown in Figure 1.7. At time $t = 0$, the user resides within the communication range of AP1 and connect to it using the wireless medium. Further, the user moves away from AP1 and reaches the vicinity of AP2 at time $t = 1$ as shown in Figure 1.7(b). During the transition from AP1 to AP2, especially when the user travels through the void between AP1 and AP2's coverage regions, it may happen that some packets destined to the user arrive at AP1 and get dropped due to the absence of user. In such cases, packets can be intelligently routed

to AP2 and buffered, if the user mobility can be predicted in advance, so that they can be delivered the moment when the user enters the communication range of AP2. However, the existing SDN infrastructures provide options only for controlled forwarding as well as dropping of packets as part of the flow-control, and not for the controlled buffering of packets at the switches. With the advantage of out-of-band control in Software Defined WLANs (SD-WLANs), such a controlled buffering has significant potential to improve the network performance in campus, office, and enterprise environments, especially while the users are on the move.

1.3.2 Disruption-prone Wireless Mesh Networks

Wireless mesh networks are characterized by the presence of one or more paths between any pair of nodes, thereby, providing robustness and reliability from node failures and link disruptions. Due to their advantages such as easy deployment, reduced infrastructure cost, and improved reliability, WMNs become the popular mode of network deployment in emergency response and tactical environments. In practice, WMNs can be used in two ways: (*i*) WLANs with wireless mesh backbones, similar to Figure 1.7 with stationary WMN replacing the wired backhaul, and (*ii*) end-user devices themselves acting as WMN routers (relay nodes) for others' packets. However, WMNs are prone to link disruptions irrespective of the nature of node mobility. For instance, Figure 1.8 shows the behavior of link disruption durations from IIST MeshNet testbed [20] involving 32 WMN routers, running Optimized Link State Routing (OLSR) protocol, under the three mobility models: (*i*) *stationary*, where all nodes remain stationary at their positions, (*ii*) *group*, where 22 nodes move as a group in the experimental zone, and (*iii*) *random*, where 22 nodes are allowed to move at their discretion.

In case of WMN backhauls, the nodes that form the mesh remain stationary and pro-

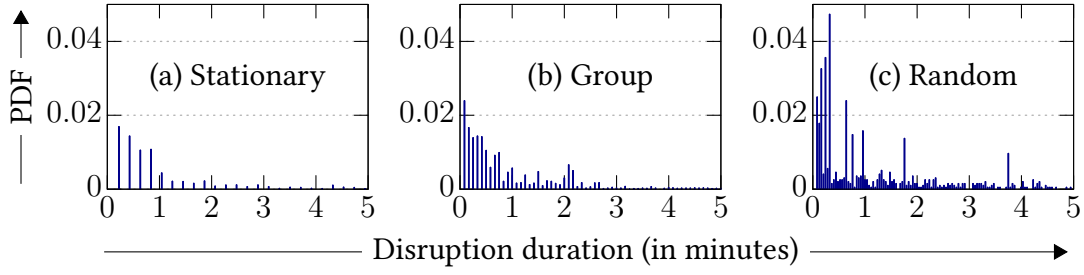


Figure 1.8: Distribution of link-disruption durations, represented as Probability Density Function (PDF), in a campus-based IISTMeshNet WMN testbed involving 32 nodes.

vide a backbone network for others to communicate [21–23]. However, such a WMN backbone is also prone to link disruptions (see Figure 1.8(a)), even with stationary nodes, due to the changing environmental conditions such as buildings, interference from other communication devices and machines, and weather. On the other hand, in mobile environments, end-user nodes take the additional responsibility of WMN routers to relay others' packets. Therefore, the mobility of such WMN routers results in significant number of link disruptions compared to a network with stationary backbone as can be seen from Figures 1.8(b) and 1.8(c) under group and random mobility models, respectively.

Even in the presence of link disruptions in wireless mesh environments, the network performance can be improved using intelligent network control based on global network information. For example, consider a WMN shown in Figure 1.9, where node u wants to communicate with node w. The time for each link to carry the data is assumed to be 1 second. At time $t = 0$, the shortest path (with a cost of 2 seconds) is available via $u \rightarrow v \rightarrow w$. At $t = 1$, the link (v, w) gets disrupted and the packets are re-routed through a long alternate path $u \rightarrow x \rightarrow y \rightarrow z \rightarrow w$ costing 4 seconds. Also, the packets already reached node v need to travel through the path $v \rightarrow u \rightarrow x \rightarrow y \rightarrow z \rightarrow w$ incurring a delay of 5 seconds. However, at time $t = 2$, the link (v, w) gets re-established. If we can anticipate the behavior of link (v, w), the packets can be buffered at node v during the link disruption and forwarded when the link gets re-established. In fact, the packets can reach node w in 3 seconds with buffering, i.e., delay through link (u, v) + delay of link disruption + delay through link (v, w), instead of ≥ 5 seconds along the alternate path. More importantly, buffering can be considered even in the absence of an alternate path, a frequent phenomenon in DTNs.

Apart from link disruptions, there exist other factors in WMNs which can affect the

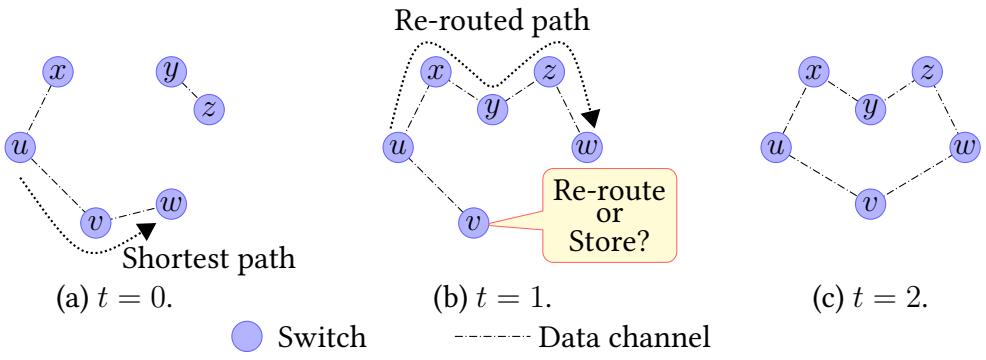


Figure 1.9: An example WMN involving link disruptions. The time (cost) required for each link to carry the packets is assumed to be 1 second.

network performance. For instance, the alternate path may get overloaded due to the re-routed flow, especially when the path is already exploited by elephant flows [24–26]. Moreover, it is revealed in [22] and [27] that more number of hops degrades the throughput as well as end-to-end delay. In fact, authors of [21, 28] and [29] showed that the number of wireless network interfaces a WMN node uses and the channels they operate have significant influence over the performance attributes. On the other hand, the problem becomes more complex in the absence of alternate paths whereby the source node gradually reduces the sending rate and subsequently leads to zero communication. Therefore, it is inevitable to have the network control with fine granularity and buffering to achieve maximum network performance, especially for the latency-aware next generation wireless networks.

1.4 The Focus and Contributions of the Thesis

The primary objective of the thesis is *to design and develop a Software Defined Disruption Tolerant Networking (SD-DTN) framework for tolerating link disruptions in software defined wireless environments, thereby, improving the network performance.* The major contributions of the thesis while achieving the objective are as follows:

1. A *self-configuration scheme* for SDN switches and controllers, to adapt with the changing wireless network characteristics, consisting of the following:
   (a) A *Software Defined Optimized Link State Routing* (SD-OLSR) protocol for SDN resource discovery in software defined wireless environments.
   (b) Two novel controller handoff schemes, *Controller-Initiated Handoff* (CIH) and *Switch-Initiated Handoff* (SIH), enabling switches to connect to the appropriate controller for achieving the required network control.
   (c) Detailed analysis of the self-configuration scheme in a Software Defined Wireless Mesh Network (SD-WMN) testbed using seven performance metrics with different cases including (*i*) switch mobility, (*ii*) controller mobility, and (*iii*) controller failure.
2. A Software Defined Disruption Tolerant Networking (SD-DTN) framework with the introduction of *controlled buffering capability* for SDN switches.
3. A mathematical model for wireless networks involving link disruptions using temporal graphs and Markov chains.

4. An *Earliest Arrival Path with Minimal Storage Time* (EAPMST) controller algorithm for SD-DTNs.

5. Design and prototyping of an SDN switch with controlled buffering capability for SD-DTNs.

   (a) Detailed performance analysis of the proposed buffering scheme using the new SDN switch prototype in an SD-DTN testbed with the three network control schemes: (*i*) benchmark SDN, (*ii*) SD-DTN, and (*iii*) EAPMST algorithm, under the three mobility models: (*i*) stationary, (*ii*) group, and (*iii*) random.

1.5 Organization of the Thesis

The thesis is organized into six chapters as follows:

Chapter 1 starts with a discussion on wireless vision emphasizing the importance of user mobility from the perspective of next generation wireless networks and the need for a flexible network control. The chapter introduces the concept of software defined networking with related components and the terms required for this thesis. Further, the chapter discusses the challenges in adopting SDN in wireless environments, especially the link disruptions. The motivation behind our research on tolerating link disruptions in software defined wireless environments is explained with examples. The chapter then defines the thesis focus and enumerate the major contributions. Finally, the organization of the thesis is outlined followed by a chapter summary.

Chapter 2 discusses the existing work related to the adoption of SDN in wireless environments. The related work in handling link disruptions in SD-WNs are divided into two sections: (*i*) handling link disruptions in control channel and (*ii*) handling link disruptions in data channel. Along with link disruptions, the chapter covers other challenges involved in SD-WNs including user mobility, controller failures, dynamic network topology, control channels in both out-of-band and in-band mode, and traffic engineering, and the proposed solutions to deal with the obstacles.

Chapter 3 deals with the link disruptions in SDN control channel and proposes a self-configuration scheme for SDN switches and controllers to adapt with the network dynamics. Two controller handoff algorithms, Controller-Initiated Handoff (CIH)

and Switch-Initiated Handoff (SIH), are designed for switches to connect with appropriate controller for network control decisions. In addition, the evolution and design of Software Defined OLSR protocol for SD-WNs are discussed in detail. Finally, the performance of self-configuration scheme is tested using an SD-WMN testbed with seven performance metrics.

Chapter 4 focuses on modeling and designing the Software Defined Disruption Tolerant Networking (SD-DTN) framework for handling link disruptions in data plane using the proposed SDN-controlled buffering capability of the switches. A new action STORE is defined along with the prototyping of an SDN switch for realizing the controlled buffering. For optimizing the end-to-end delay and buffering parameters, an Earliest Arrival Path with Minimal Storage Time (EAPMST) controller algorithm is designed. The chapter analyzes the performance of the proposed controlled buffering scheme and EAPMST under three mobility models on a disruption-prone SD-WMN testbed.

Chapter 5 provides our contribution toward the emerging applications of SD-DTNs including (*i*) software defined vehicular networks, (*ii*) software defined satellite networks, (*iii*) software defined tactical networks, and (*iv*) software defined emergency response networks.

Chapter 6 concludes the thesis by enumerating the major conclusions and the directions for future research.

1.6 Summary

In this chapter, we discussed the wireless vision and the importance of handling link disruptions to achieve the Quality-of-Service (QoS) requirements of next generation communication infrastructures. The need for a flexible network control is outlined with SDN as the suitable candidate for achieving programmable networks. An introduction to the concept of SDN is provided along with the challenges in adopting the concept in wireless environments. The motivation for our work is briefed using example networks involving link disruptions. Further, the primary focus of the thesis is defined and the major contributions are enumerated. The organization of the thesis is outlined with the focus of each chapter. In Chapter 2, we discuss the existing work related to software defined wireless networks emphasizing the challenges in both SDN control channel and data plane, and different approaches adopted to resolve them.

Chapter 2
Related Work

> "*Bernard of Chartres used to compare us to [puny] dwarfs perched on the shoulders of giants. He pointed out that we see more and farther than our predecessors, not because we have keener vision or greater height, but because we are lifted up and borne aloft on their gigantic stature.*"
>
> — John of Salisbury

The idea of separating control plane from the data forwarding plane for realizing programmable networks gained its momentum during the early 1990s, i.e., during the beginning of Internet revolution. In [30], Feamster et al. divided the evolution of programmable networking into the three following phases: (*i*) active networks, where a programming interface is provided to the data plane devices either via in-band, called capsule module, or via out-of-band, called programmable switch/router model, (*ii*) separating control plane from data plane, where an interface is provided for the logically centralized control plane to program the network devices in data plane, and (*iii*) OpenFlow Application Programming Interface (API) and network operating systems (known as controllers). OpenFlow protocol [11] provides an interface for the controllers to communicate the network control decisions (i.e., flow-rules) with the data plane elements in order to achieve the desired network control.

As far as wireless networks are concerned, authors of [31] and [32] surveyed the adoption of SDN concepts in different environments such as cellular networks, wireless sensor networks, wireless mesh networks, home networks, wireless LANs, Mobile Ad hoc Networks (MANETs), and vehicular networks. Software Defined WLANs (SD-WLANs) are separately surveyed by Dezfouli et al. in [33] where the uncertainty of physical layer features, especially related to the wireless medium, is identified as the most important challenge in adopting SDN in wireless networks. Such an uncertainty is the result of shared

communication medium, interference, and user mobility. Both the variations in link quality as well as the link disruptions themselves prompt corresponding changes in the global network view and necessitate the controller to re-compute the flow-rules very frequently. The re-computed flow-rules should reach the switches in minimum time to immediately take-effect the network change to the existing flows. However, it may happen that the network undergoes another state-change when the rules are on their way, thereby, making the recent flow-rule invalid. Besides such a delay in switch-controller communication, the attributes of flow-rules, as studied in [34], also have significant influence over the load on the control channel, load on the controller, and the memory overhead on the switches. Therefore, SD-WNs necessitate research in the architectural level as well as in the control and data plane level to efficiently address the above-mentioned challenges.

2.1 Handling Disruptions in SDN Control Channel

In any SDN, it is important for the controller to deliver the flow-rules to the switches in order to achieve the required flow control. In next generation communication infrastructures demanding extremely low latency requirements, such flow-rules should reach the switch within minimum time upon a PacketIn request. However, the uncertainties in the wireless medium that forms the SDN control channel offer a major hurdle in delivering the flow-rules within the required time. The problem becomes more complex when the control channel operates in in-band mode where the data and control packets share the same wireless infrastructure. In addition to the delay in control channel, frequent link disruptions lead to the failure of switch-controller path and, as a result, the switches get disconnected from the controller compromising the desired network control. In order to prevent the situations of such controller unavailability, SDNs make use of multiple physical controllers in the control plane [35]. However, identifying the locations for placing the controllers, known as the Controller Placement Problem (CPP) [36–39], in such a way to minimize the switch-controller delay is an important concern in both wired as well as wireless networks. Moreover, the solutions proposed for CPP assume a consistent availability of control channel between the switches and controllers. Such an assumption becomes invalid in wireless environments involving mobile switches and controllers as in software defined vehicular networks, software defined satellite networks, and software defined tactical and emergency response networks. In this thesis, we address the inconsistencies in SDN control channel using a novel resource discovery and self-configuration scheme for switches and controllers to adapt with the dynamically changing network.

Besides deploying multiple physical controllers, different methods were adopted in literature to reduce the latency involved in switch-controller communication. As mentioned in Section 1.2.1, out-of-band control is the preferred choice for networks with fixed backbone while in-band control forms the only feasible option in networks that involve mobile switches and controllers. An SDN architecture for WMNs was designed in [40] where separate sub-bands were allotted for control and data traffic using any of the following three algorithms: (*i*) Fixed-Bands Non-Sharing, (*ii*) Non-Fixed-Bands Non-Sharing, and (*iii*) Non-Fixed Bands Sharing. The non-sharing methods use out-of-band control while the sharing method employs in-band control. As far as SD-WMNs involving link disruptions are concerned, the use of a single network interface in mesh routers, interference in wireless medium, network congestion, and the extraction of global network view are considered as major challenges in achieving the SDN objectives. In [41], Dely et al. designed a framework integrating OpenFlow with WMNs using out-of-band control by equipping mesh routers with multiple network interfaces. A dedicated monitoring and control server is used to extract the topology related information from the OpenFlow mesh routers. Experimental results suggested that keeping a small number of flow-rules improves the network performance due to the reduced search complexity while the rule activation time increases proportional to the network load. Further, authors of [41] demonstrated client mobility management in SD-WMNs with controlled handovers.

Emphasizing the importance of switch-controller communication, authors of [42] proposed a three-stage routing architecture for SD-WMNs. In the first stage, the switches identify their paths toward the controller and the selected paths are optimized in the second stage. In the third stage, the routing decisions are sent to the mesh nodes through the optimal path incurring minimum switch-controller delay. A different approach was followed in DIFANE (expanded as Define It Fast ANd Easy) [43], where the flow-rules get cached in a subset of switches, known as *authority switches*, and the packets are intelligently routed through a set of selected switches in order to achieve the flow control. Here, only the authority switches are entitled to request flow-rules from the controller while other ingress switches request authority switches for the flow-rules. Similar to DIFANE, Curtis et al. suggested DevoFlow in [44] allowing the switches to compute the flow-rules for micro-flows while the controller governs the elephant flows. Simulation results showed the efficacy of both DIFANE and DevoFlow in reducing the SDN control traffic to a great extend, thereby, improving the network throughput.

Apart from the problems due to the delay incurred in switch-controller communication, the controller may get disconnected from the switches during network partitioning,

resulting from the mobility of nodes. In order to handle such disconnections, Detti et al. proposed a wireless mesh Software Defined Network (wmSDN) architecture operating under in-band control [45]. In wmSDN, the switches and controllers run Optimized Link State Routing (OLSR) protocol [46] for providing a platform for the in-band control channel. Each switch in wmSDN is equipped with an OLSR-to-OpenFlow (O_2O) module to map the OLSR's topology information to corresponding flow-rules in flow-tables during emergencies such as controller failure or controller disconnection. Similar to [45], Nguyen-Duc and Kamioka proposed an extended controller architecture in [47] with a controller module in switches to handle controller failure/disconnection. Yeganeh and Yashar conceived the switch-controller delay and the inability of controller to handle high network load as design issues rather than considering them as problems inherent to SDN. Therefore, a new software platform was suggested in [48] that focuses on the optimal placement of network functions, maximum resource utilization, and easiness in use.

Besides integrating limited controller capability to the switches, the use of multiple physical controllers is considered as a feasible option to handle controller disconnections. That is, multiple controllers can be deployed at specific locations in such a way that each switch gets access to *at least* one controller for the flow-rules. In [20], Mamidi et al. designed controller handoff schemes for switches to connect to the nearest controller using two metrics, Round Trip Time (RTT) and Expected Transmission Count (ETX). Here, a master controller is assigned with the task of determining appropriate controller for the switches using the information from other controllers sent via a newly defined east-west (i.e., controller-to-controller) interface. The experimental results showed that ETX forms a stable metric for controller handoffs compared to RTT in avoiding handoffs due to sheer fluctuations in the handoff metric. Since the use of multiple controllers provides an additional benefit of load sharing among the controllers, Dixit et al. proposed a framework called ElastiCon in [49], where each switch is integrated with a migration module to change its controller based on the controller load. While [20] and [49] emphasized on switch-controller interaction, authors of [50] used multiple controllers for handling the mobility of end-users. Here, global controllers are entrusted to handle the flows between different domains while local controllers govern the network traffic within a domain. In multi-controller paradigms, especially with fixed backhaul, researchers considered the controller placement problem as an important concern and proposed various solutions in order to improve network scalability, robustness, and resilience from controller failures [36–39, 51–54]. However, such backhaul-based solutions for CPP, assuming a consistent SDN control channel, become void in wireless environments involving highly mobile

switches and controllers. In this thesis, we try to address the problem with a resource discovery scheme for SDN switches and controllers to self-configure themselves depending on the dynamic network behavior.

2.2 Handling Disruptions in SDN Data Plane

Software defined networking is considered as the panacea for issues emanating from the dynamism in data plane characteristics, especially the ones envisaged in next generation communication networks. With an assurance of global network view and a reliable communication channel between control and data planes, SDN provides the options to control the flows depending on the changing network behavior, thereby, improving the network performance compared to the existing wireless routing protocols [55]. As far as the link disruptions in data plane are concerned, Disruption Tolerant Networking (DTN) becomes the preferred choice for disruptions lasting long durations while rate adaptation and traffic re-routing form feasible candidates for disruptions of short durations.

In DTNs [15], nodes use a *store-carry-and-forward* approach to route and deliver the data to the destination. A separate layer called *bundle layer* running a *bundle protocol* [16] is integrated between the transport and application layers of the TCP/IP protocol stack to realize the store-carry-and-forward operation. Further, different DTN routing protocols were designed with the objective of maximizing the delivery ratio while minimizing end-to-end delay. Examples of DTN routing protocols include epidemic protocol [56], MaxProp [57], PRoPHET [58], and Spray and Wait [59]. As far as SDNs in disruption-prone environments are concerned, Zacarias et al. proposed an SDN-based solution in [60] for tactical edge networks to address the DTN-related issues arising from resource constrained nodes such as user equipment, UAVs, and sensor nodes. Here, the controller is equipped with a DTN orchestrator to manage the store-carry-and-forward operation. An SDN framework, called SERvICE (expanded as Software dEfined fRamework for Integrated spaCe-tErrestrial satellite communication), for satellite-terrestrial networks was proposed by Li et al. in [61]. In SERvICE, the SDN control channel is operated over a DTN formed over the mobile satellites. Similar to [61], an SDN framework for satellite networks was suggested in [62] where the control channel is operated over a bundle layer tunnel. In general, DTN routing protocols generate multiple copies of the same message to achieve fast delivery by routing them through multiple paths. However, multiple message copies result in significant communication overhead on the entire network. If the mobility pattern of nodes can be predicted in advance, the overhead can be significantly reduced

with minimum number of message copies by intelligently buffering them at intermediate nodes during link disruptions and routing them when the links become alive.

As far as the link disruptions of short durations are concerned, multipath routing, traffic re-routing (as shown in Figure 1.9), and rate adaptation are considered as the most viable options to achieve the required throughput in non-SDN networks. In fact, SDN offers flexibility to the above three techniques so that administrators can control the network flows depending on changing user requirements and network dynamics rather than taking decisions in a deterministic fashion. In [63], Fu and Wu investigated the feasibility of an SDN-based disjoint multi-path routing scheme. The scheme is tested on complete and grid network models, and the results showed an improvement over the traditional shortest-path-based routing techniques. Due to the crucial role played by the elephant flows in deciding the network performance, authors of [26] designed an elephant flow detection method followed by an SDN-based multipath routing scheme for such flows. Supplementing the ideas of [26] and [63], Schepper et al. realized a multipath TCP in [64] exploiting the flexibility provided by the SDN framework. Further, the route selection in [65] focused on balancing the network load as well as reducing the controller overhead.

The technique of rate adaptation was adopted in [66] by varying the attributes of TCP, known as Dual-mode TCP, to deal with delays/disruptions in the network. An Integer Linear Programming (ILP)-based approach, called Ant Colony Online Flow-based Energy efficient Routing (AC-OFER), was proposed in [67] for flow re-configuration and rate adaptation depending on the user demands, specifically for wireless environments characterized by random arrival and departure of users. An abstraction called *transmission policy* was suggested in [68] for re-configuring the rate control between clients and wireless access points to resolve user mobility in enterprise WLANs. Even though the above-mentioned schemes provide improvements in performance over the existing methods in their own respect, a combination of rate adaptation and multipath routing is able to obtain better results as shown in [69].

In the context of SDN-based network management in wireless domain, authors of [45, 70–72] considered networks equipped with wireless mesh backhaul. The performance of popular wireless mesh routing protocols, such as Destination-Sequenced Distance Vector (DSDV), Ad hoc On-demand Distance Vector (AODV), and OLSR, were evaluated under SDN-controlled mobility environments. Bera et al. suggested Mobi-Flow [73], a Mixed Integer Linear Programming (MILP)-based model for dynamic flow-rule placement at the switches depending on the mobility pattern of users. On the other hand, authors of [74]

proposed an MILP model to optimize access point association, gateway selection, and backhaul network routing in SD-WMNs. Even though the traffic engineering techniques such as traffic re-routing, multipath routing, and rate adaptation provide performance improvements, such techniques may fail in the absence of an alternate path or the existing alternate paths are constrained due to factors such as limited bandwidth, network congestion, and the presence of elephant flows. In this context, our SD-DTN framework offers an SDN-based controlled buffering capability, along with the re-routing options, for switches to buffer the packets at intermediate nodes during disruptions and forward them as and when a path is established toward the destination.

Due to the flexibility of fine grained flow-control, SDN is considered as one of the promising candidates for solving many possible problems in next generation wireless networks. For example, in [75], Parvez et al. surveyed the 5G-based low-latency aware networks where the end-to-end latency incurred in a communication session is split into the following three stages: (*i*) Radio Access Network (RAN), (*ii*) core network, and (*iii*) caching. Also, Network Function Virtualization (NFV), where SDN is the major component, forms the backbone of the core network in meeting the delay constraints. Further, authors underlined the role of radio parameters in improving the network performance. Following [75], Coronado et al. proposed a framework 5G-EmPOWER in [76] to capture the dynamism in wireless networks, especially in 5G RANs. A new southbound interface called OpenEmpower protocol was designed for 5G-EmPOWER as an alternative to the traditional OpenFlow protocol. On the other hand, authors of [77] proposed an SDN-based wireless backhaul called SODALITE for 4G and 5G networks retaining OpenFlow protocol as the southbound interface. Moreover, authors of [78] proposed an architecture, named as WiSA, focusing on future wireless systems. In WiSA, the OpenAir module controls the radio parameters while OpenCore manages the QoS requirements of upper layers. Use of heterogeneous infrastructures within a single communication session is considered to be one of the future directions in achieving the low-latency requirements of next generation communication paradigms. For instance, authors of [79] proposed an SDN-based approach in offloading the data traffic from cellular networks to WLANs near the user's vicinity, while a branch and bound-based algorithm was designed by Das et al. in [80] for improving the utility of wireless domain where the flows involve wired as well as wireless infrastructures.

In short, any effort in improving the performance can play a crucial role in achieving the objectives of future communication systems. In this context, our proposed SD-DTN framework offers a value addition to the existing as well as next generation communi-

cation infrastructures with an additional controlled buffering capability. Such an SDN-based buffering can be a way forward to handle link disruptions in networks equipped with wired and wireless backhauls (as discussed in Section 1.3.1 and Section 1.3.2, respectively), as well as at the last mile.

2.3 Summary

In this chapter, we discussed the existing research in handling the issues emanating from link disruptions while adopting SDN into wireless environments. The chapter is divided into two sections: (*i*) handling link disruptions in control channel and (*ii*) handling link disruptions in data plane. The primary issue in any SD-WN is in maintaining the SDN control channel in order to provide the network control decisions (flow-rules) to the switches at the earliest. Therefore, we first discussed the attempts in resolving issues in SDN control channel such as deployment of multiple controllers and the delegation of controller roles to the switches in handling controller failures and disconnections. On the other hand, rate adaptation, multipath routing, and DTN approaches were the focal points of discussion in Section 2.2 on handling link disruptions in data plane. A detailed classification of existing approaches toward disruption-prone software defined wireless networks is depicted in Figure 2.1. As far as link disruptions in SD-WNs are concerned, authors restricted their focus only to disruptions arising from the mobility of end-users. However, next generation communication systems envision mobile switches and controllers in order to achieve fine-grained network control. Therefore, we consider software defined wireless environments with mobile switches and controllers, and design a self-configuration scheme in Chapter 3 for resolving challenges in SDN control channel. Further, we introduce an SD-DTN framework with controlled buffering capability in Chapter 4 to handle link disruptions in data plane, thereby, improving the network performance.

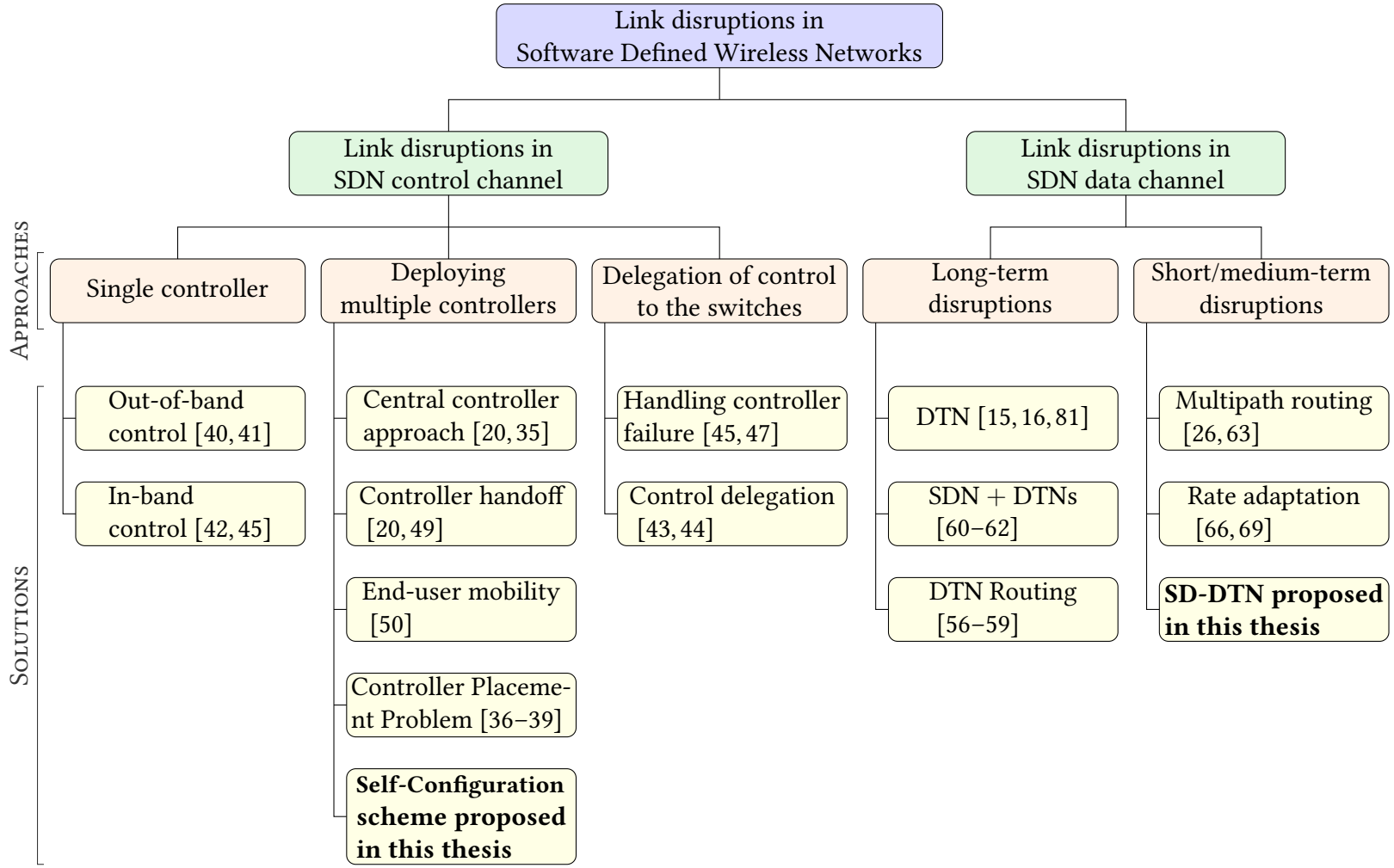


Figure 2.1: Classification of link disruptions in software defined wireless networks and the proposed solutions in literature for handling them.

Chapter 3

Self-Configuration in Software Defined Wireless Networks

> "*A moment of disruption is where the conversation about disruption often begins, even though determining that moment is entirely hindsight.*"
>
> — Steven Sinofsky

The concept of software defined networking, originally developed for experimental wired networks, finds its immediate application in campus/enterprise networks [82–87] as well as in high-traffic data centers [88–96]. Moreover, the fine-grained network control that SDN offers makes the concept a suitable candidate for solving the issues in next generation wireless networks [2, 40, 79, 97–106]. The adoption of SDN to wireless domain opens up a number of challenges and requires a redesign of the existing SDN architecture to deal with uncertainties in the wireless environment. In this chapter, we focus on the issues in SDN control channel, especially the link disruptions due to the mobility of SDN switches and controllers. We propose a state-of-the-art SDN resource discovery and self-configuration scheme for switches and controllers to make the network adjust depending on the network dynamics.

In SDNs with fixed backhaul, a flow-rule remains in the flow-table for significantly long time unless there exists a need to re-route the packets due to emergency situations such as a node failure or the resurgence of elephant flows [107]. As discussed in Section 1.1, a flow-rule is essentially a function of the global network view that includes the network topology, network traffic, and the attributes of data plane devices. In wireless environments involving mobile devices, the link quality varies more often and results in frequent topology changes. Therefore, the controller needs to re-compute and update the flow-rules in each switch according to the modified global network view. Similar to the

data channel, SDN control channel is also subject to frequent link disruptions, which make the controller unavailable for switches, thereby, compromising the objectives of SDN. The problems due to link disruptions become more complex when the networks operate under in-band control where the data and control packets share the same wireless infrastructural resources and channels.

Deploying multiple controllers is considered as a feasible option to deal with controller disconnections and controller failures. However, the placement of controllers becomes a crucial factor in maximizing the network performance besides providing resilience against controller failures. As discussed in Section 2.1, the existing controller placement algorithms are designed for networks involving stationary nodes (both switches and controllers). Such algorithms become invalid in mobile ad hoc networks where new nodes get added and existing nodes leave the network in real-time. That is, SDN requires a real-time controller selection/assignment scheme for the switches in order to connect to the best controller for obtaining the flow-rules at the earliest. Therefore, we propose a self-configuration scheme with the following options:

1. Multiple controller support to handle controller failures and disconnections.

2. Two controller handoff schemes, Controller-Initiated Handoff (CIH) and Switch-Initiated Handoff (SIH), to make the switches connect to the appropriate controller with minimum user intervention.

3. A Software Defined Optimized Link State Routing (SD-OLSR) protocol, works under in-band control, for automated SDN resource discovery (which is used in SIH) as well as for capturing the global network topology.

For the remaining of our discussion, we follow the OpenFlow terminologies described in Table 3.1 for different types of SDN control packets. In addition, we use the following assumptions for our network model:

1. Each network node is equipped with a single Network Interface Card (NIC).

2. Each SDN switch in the network is connected to the control plane through at least one controller, i.e., there exists at least one controller to serve each network partition.

3. All controllers possess the same global network view.

4. The SDN control channel is operated in in-band mode with switches and controllers running the Optimized Link State Routing (OLSR) protocol.

5. All switches are connected to all available controllers in Controller-Initiated Handoff.

Table 3.1: Different types of SDN control messages used in OpenFlow.

Message Type	Size (bytes)	Direction	Description
EchoRequest	74	S⟶C	Checks the availability of controller.
EchoReply	74	C⟶S	Acknowledgment for the controller's presence.
FeaturesRequest	74	C⟶S	Queries the features of a switch.
FeaturesReply	98	S⟶C	Description of switch's attributes.
MultipartRequest	82	C⟶S	Queries the status of individual flows from a switch.
MultipartReply	210	S⟶C	Description of the switch's flow-rule status.
PacketIn	Variable	S⟶C	On a flow-rule miss or on a CONTROLLER action (i.e., a flow-rule request).
PacketOut	Variable	C⟶S	Custom packets to the switches.
FlowMod	Variable	C⟶S	Modify-State message with a flow-rule to change the state of a switch.
RoleRequest	90	C⟶S	Assign/query controller's role to/from a switch.
RoleReply	90	S⟶C	Role of the controller toward the switch.

**Note:* C⟶S denotes Controller-to-Switch and S⟶C denotes Switch-to-Controller.

3.1 Controller-Initiated Handoff

In Controller-Initiated Handoff (CIH), we make use of the controllers to take decisions, based on the global network view, on which controller each switch should contact to obtain the flow-rules with minimum delay. We design CIH based on the basic construct of SDN architecture that employs multiple controllers. In SDNs equipped with multiple controllers, each switch needs to be configured to connect to all available controllers at the start-up. The relationship of a controller toward a switch is attributed using the following three roles:

1. **Primary:** A controller holding a *primary* role toward a switch has complete control over that switch, i.e., the controller can *query and modify the state* of that switch. Therefore, switches send PacketIn messages to their *primary* controller for the required flow-rules. From the perspective of data plane, a switch can hold only one *primary* controller at a time. When a switch is assigned with a new *primary* controller, the existing *primary* controller changes its role to *secondary*[1].

2. **Secondary:** A controller with a *secondary* role toward a switch can *only query the state* of that switch and not permitted to modify the state. Also, multiple controllers can possess the *secondary* role toward a switch at the same time. In other words, switches do not contact *secondary* controller(s) for the flow-rules.

3. **Equal:** An *equal* role is same as that of a *primary* role (i.e., complete rights over the switch) except the fact that a switch is allowed to possess multiple controllers with *equal* role at the same time. Upon a flow-rule miss, a switch sends PacketIn messages (i.e., flow-rule requests) to all controllers possessing the *equal* role. However, such multiple PacketIn messages and corresponding FlowMod responses create additional overhead on the network.

At the time of an initial switch-controller connection setup, the controller assumes *equal* role toward the switch by default. Further, the role of each controller is assigned depending on the SDN controller application responsible for the role selection. In short, the controller selection for a switch becomes a *role-assignment problem* provided that *each switch has an established connection with all available controllers*. Besides selecting a controller as *primary* for a switch, it is required to make other controllers from *equal* or *primary* to the *secondary* role. The change of role from *equal* to *secondary* aims at reducing the redundant traffic generated by multiple PacketIn-FlowMod request-response packets. In order to resolve the role-assignment problem, we propose a controller-initiated handoff algorithm to select the appropriate *primary* controller for each switch. Since the mobility of switches and controllers results in a wireless mesh environment, we use the Optimized Link State Routing (OLSR) protocol to provide in-band control channel for SDN due to its low control overhead. Figure 3.1 shows the SDN architecture for CIH with controller handoff module integrated with the controller.

For finding the *primary* candidate among multiple controllers, we use the metric Expected Transmission Count (ETX) [108] used by the OLSR protocol. The ETX of a link

[1]In SDN literature and specifications, the *primary* and *secondary* roles were referred as *master* and *slave*, respectively.

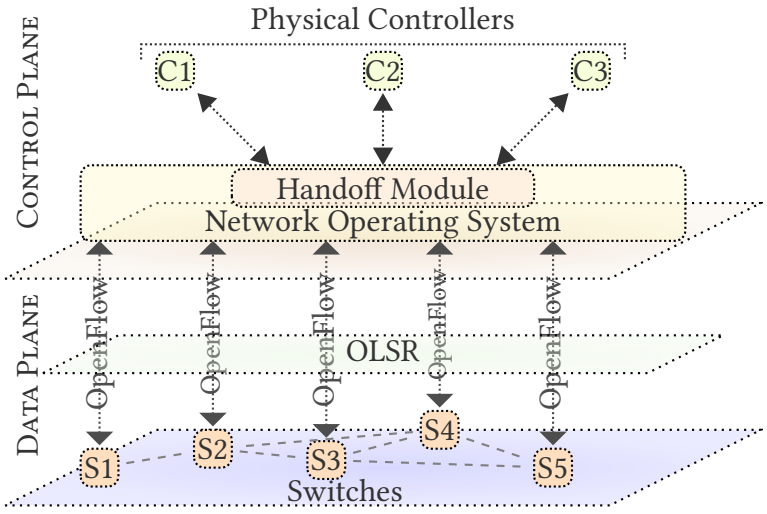


Figure 3.1: SDN architecture for controller-initiated handoff.

(u, v) is defined as the number of transmissions required to successfully transfer a packet from node u to v. We observe in [20] that ETX offers a stable metric over a specific period of time compared to the metric Round-Trip Time (RTT) in terms of avoiding handoffs due to sudden fluctuations in the values of the handoff metric. In addition, the isotonicity property [109] of ETX helps to compute the cost over a multihop path connecting switches and controllers, which is a necessary requirement for software defined WMNs. In CIH, each controller is assumed to have the global network view, provided by the OLSR protocol, for selecting the *primary* controller for the switches. Besides the requirement that each switch needs to have an established connection with all controllers, in CIH, each controller should be provided with the information of other controllers, i.e., a controller in CIH computes the cost of all switches toward all available controllers and selects the switches to which it should serve as the *primary* controller. Upon the failure of a controller, the cost of switches reaching that controller becomes infinity, thereby, making it the farthest controller. Algorithm 3.1 describes the procedure followed by controller C in CIH.

In CIH, each controller computes the cost of all switches toward all available controllers. Algorithm 3.1 considers an instance of CIH at controller C at time t. In Line 2, controller C computes the cost of switch s in reaching all controllers and selects the nearest (in terms of ETX) controller C_s^-. If controller C is not the nearest and holding an *equal* role, then the role of C is reverted to *secondary* (Lines 3–6). If controller C is found to be the nearest one, the controller checks the switch's cost toward the existing *primary* controller C_s^+. If the difference is found to be greater than a threshold τ, controller C

Algorithm 3.1: Controller-Initiated Handoff (CIH) by controller C at time t.

Data: $\mathbb{C}$ — Set of controllers
$\mathbb{S}$ — Set of switches
$\mathcal{R}_s^c$ — Role of controller c on switch s, $\mathcal{R}_s^c \in \{\textit{equal, primary, secondary}\}$
$\delta_{u,v}^t$ — Cost between nodes u and v at time t
C_s^+ — Current *primary* controller of switch s
C_s^- — Nearest controller to switch s

1 **foreach** $s \in \mathbb{S}$ **do**
2 $\quad C_s^- \leftarrow \arg\min_c \{c \mid c \in \mathbb{C} \wedge \forall x \in \mathbb{C} : \delta_{c,s}^t \leq \delta_{x,s}^t\}$
3 $\quad$ **if** $C_s^- \neq C$ **then**
4 $\quad\quad$ **if** $\mathcal{R}_s^C = \textit{equal}$ **then**
5 $\quad\quad\quad \mathcal{R}_s^C \leftarrow \textit{secondary}$
6 $\quad\quad$ **end**
7 $\quad$ **else if** $C_s^+ \neq C$ **then**
8 $\quad\quad \Delta \leftarrow \dfrac{\delta_{C_s^+,s}^t - \delta_{C,s}^t}{\delta_{C_s^+,s}^t} \times 100$
9 $\quad\quad$ **if** $\Delta > \tau$ **then** ▷ τ : Handoff threshold
10 $\quad\quad\quad \mathcal{R}_s^C \leftarrow \textit{primary}$
11 $\quad\quad$ **end**
12 $\quad$ **end**
13 **end**

takes the responsibility of *primary* controller for the switch s (Lines 7–11) and the existing *primary* controller C_s^+ reverts to the *secondary* role. The threshold τ is introduced as a tuning parameter to limit the frequent handoffs resulting from minor variations in the costs of a switch toward different controllers. Apart from τ, we use the exponential smoothing technique to compute the cost between two nodes in order to compensate for the sudden fluctuations in ETX samples, i.e., the cost $\delta_{u,v}^t$ between two nodes u and v at time t is estimated as

$$\delta_{u,v}^t = \alpha \mathcal{X}_{u,v}^t + (1-\alpha)\delta_{u,v}^{t-1}, \tag{3.1}$$

where $\mathcal{X}_{u,v}^t$ is the ETX required to successfully transfer a packet from node u to node v at time t and α is the smoothing factor. The smoothing factor allows us to tune the handoff process in terms of the fluctuations in handoff metric. For instance, higher the value of α, the more probable will be the handoff upon a sudden fluctuation in ETX. On the other hand, the threshold τ ensures the handoff to proceed only if the newly computed nearest controller is at a minimum distance away from the existing controller. From empirical observations, we chose $\tau = 10$ and $\alpha = 0.3$ for our experiments. The number of handoffs and the inter-handoff time with varying values of α and τ are detailed in Section 3.5.5.1.

Since each controller computes a switch's cost toward all other controllers, the algorithm runs in $\mathcal{O}(|\mathbb{C}|\times|\mathbb{S}|)$ time.

3.1.1 Application Scenarios for CIH

Controller-initiated handoff follows the original SDN concepts that the controller dictates the switch's behavior. Here, the controller computes the nearest controller for each switch and requests the switches to shift to appropriate *primary* controller in order to obtain flow-rules in minimum time. Since the controller handoff is realized using role-change, it is a prerequisite for the switches to have an established connection with all controllers and each controller needs to be configured with the information of other controllers as well. Therefore, in CIH, the addition of a new switch requires a manual configuration for assigning all controllers to that switch. On the other hand, the deployment of a new controller is a more cumbersome task that all switches need to establish a connection with the new controller and the existing controllers need to be notified about the presence of the new controller. Such a real-time re-configuration becomes a difficult task in cases (such as software defined wireless mesh networks) where the network involves large number of switches and controllers. Therefore, CIH is best suitable for networks where the addition of new nodes is less frequent and the re-configuration of nodes is an easy process.

In order to deal with the limitations of CIH, we propose a switch-initiated handoff scheme for networks to self-adapt with the changing network characteristics with near-zero configuration effort.

3.2 Switch-Initiated Handoff

Unlike CIH, in Switch-Initiated Handoff (SIH), the handoff procedure is initiated from the switch itself, i.e., each switch finds out the best (in terms of the minimum cost with respect to ETX) controller to request the flow-rules. However, to find out the cost toward available controllers, the switch needs to be configured with the information of all controllers. As discussed in Section 3.1.1, such a manual configuration is difficult in cases when (*i*) the network involves large number of nodes and (*ii*) the network is subject to the addition of new nodes or the removal of existing nodes in real-time. That is, an SDN resource discovery mechanism is inevitable for the switches to identify different controllers in the network. Therefore, we extend the OLSR protocol, which is used to maintain the SDN control channel, to Software Defined OLSR (SD-OLSR) to accommodate the automated

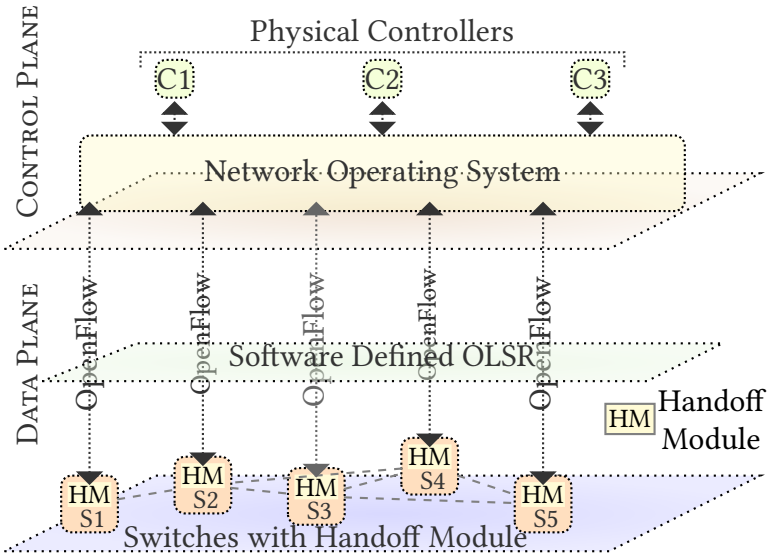


Figure 3.2: Architecture of switch-initiated handoff.

SDN resource discovery besides its role in capturing the dynamism of global network view. A detailed description of the design of SD-OLSR protocol is provided in Section 3.3.

Since OLSR is a link-state routing protocol, each node in the network possesses the entire network topology or the topology of the network partition where the node resides. In SIH, we enable the SD-OLSR protocol to provide the network topology *along with the type of each node*, i.e., a controller, a switch, or both. Figure 3.2 shows the architecture of SDN for SIH with a controller Handoff Module (HM) at each switch. Upon constructing the network topology from SD-OLSR, the switch selects the appropriate controller as per Algorithm 3.2. Similar to CIH, the handoff module of the switch makes use of the exponential smoothing function provided in Eqn. (3.1) to compute the cost toward available controllers, thereby, enabling the tuning of handoff process depending on the application environment. The switch s computes the shortest path (in terms of the cost derived from ETX) toward each controller and selects the nearest one (Line 1). If the nearest one is different from the existing controller (Line 2), the algorithm checks for the threshold τ to proceed with the handoff process (Line 3), i.e., if the difference is found to be greater than the threshold, the switch selects its new controller (Lines 4 and 5). When a controller fails, the Topology Control packets of OLSR protocol do not reach the switches, thereby, making the controller infinite distance away from the switches. As a result, the switches shift to the nearest controller among the other active controllers. Since the switch computes the shortest path only toward the available controllers, the algorithm runs in $\mathcal{O}(|\mathbb{C}|)$ time.

Algorithm 3.2: Switch-Initiated Handoff (SIH) by switch s at time t.

Data: $\mathbb{C}$ — Set of controllers
$\mathbb{S}$ — Set of switches
$\delta^t_{u,v}$ — Cost between nodes u and v at time t
C^+_s — Current controller of switch s
C^-_s — Nearest controller to switch s

$C^-_s \leftarrow \arg\min_c \{c \mid c \in \mathbb{C} \wedge \forall x \in \mathbb{C} : \delta^t_{c,s} \leq \delta^t_{x,s}\}$
if $C^+_s \neq C^-_s$ **then**
  $\Delta \leftarrow \frac{\delta^t_{C^+_s,s} - \delta^t_{C^-_s,s}}{\delta^t_{C^+_s,s}} \times 100$
  if $\Delta > \tau$ **then** ▷ τ : Handoff threshold
    $C^+_s \leftarrow C^-_s$
  end
end

3.2.1 Application Scenarios for SIH

In SIH, the switches establish connection only with the nearest controller, *not with all controllers* as required in CIH. Such a single connection per switch reduces the control overhead to a significant extent. In addition, SD-OLSR protocol makes the controller information available to the switches, thereby, removing the manual configuration while adding a new node (either switch or controller) or when an existing node leaves the network. Moreover, SIH is more suitable for realizing resilience from switch/controller failures, i.e., the failed node or switch can be replaced with a new node, whose information is automatically broadcasted to other nodes in the network and as a result, the other nodes get self-adjusted with the new switch/controller. One of the major limitations of CIH is in maintaining the switch-controller connections during network partitioning. Since switches in SIH focus only on the nearest controller, failure of paths toward other controllers does not affect the SDN control channel.

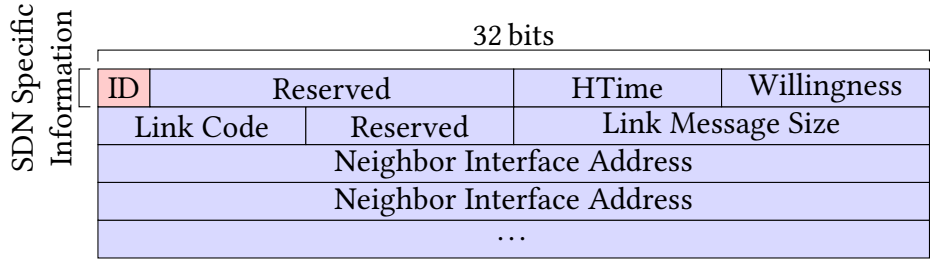
3.3 Software Defined OLSR (SD-OLSR) Protocol

Since CIH may not be always feasible in wireless environments involving switch and controller mobility, it is inevitable to equip switches with required control functions to handle the dynamic network behavior. Authors of [43, 44] and [48] also opined that delegating a set of control functions to the switches has significant influence on improving the network

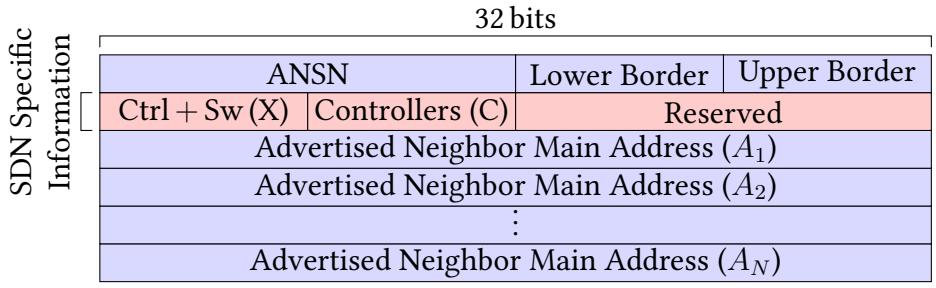
performance. In order to take such control decisions, switches also need to be assured to possess the required network information. To the best of our knowledge, Software Defined OLSR becomes the first step toward that direction for wireless environments with an SDN resource discovery scheme to help nodes to self-configure themselves with required control functions.

Optimized link state routing protocol [46] follows a link state routing approach and employs different control messages to exchange the node and link level information among all nodes in the network. The control packets generated by a node include the identity of the node (IP address), identity of its neighbors, quality of the links formed with the neighbors, and the information for sequencing and synchronization. In OLSR, each node selects a minimal set of its neighbors, known as *Multi-Point Relays* (MPRs), through which it can reach all its two-hop neighbors. The set of nodes which select node A as their MPR are known as *MPR-Selectors* of node A. Each node running OLSR sends its IP address and the information of links connected to it, to the neighbors using HELLO messages, thereby, helping the neighbor nodes to select their MPRs. Further, MPRs broadcast the information of their MPR-Selectors to the entire network using TOPOLOGY CONTROL packets. Since only MPRs send the broadcast packets, OLSR incurs minimum control overhead compared to the other WMN protocols. In order to realize automated SDN resource discovery, we make use of the HELLO and TOPOLOGY CONTROL messages of the OLSR protocol defined in [110].

In SDN, we classify the nodes into the following three types: (*i*) switch, (*ii*) controller, and (*iii*) switch + controller. In SD-OLSR, we modify the HELLO packets in such a way that a node can include its SDN-type in the packet and send to the neighbors. Besides HELLO packets, the format of TOPOLOGY CONTROL messages are also amended with the additional information to extract the SDN-node types of MPR-Selectors. The extended formats of HELLO and TOPOLOGY CONTROL messages, and the description of each message field are provided in Figure 3.3 and Table 3.2, respectively. In the HELLO packet, we make use of two bits of the *Reserved* field, represented as the field *ID* in Figure 3.3(a), to represent the type of SDN-node, i.e., 0 for switch, 1 for controller, and 2 for switch + controller. Upon receiving the HELLO packets, MPRs create a TOPOLOGY CONTROL message, where the MPR selectors are arranged in the *Advertised Neighbor Main Address* (ANMA) list in such way that the nodes with switch + controller capability (i.e., type-2) are listed first, followed by controllers (i.e., type-1), and finally, the switches (type-0). To identify the count of each SDN-node type in the ANMA list, we introduce two fields *Ctrl+Sw (X)* and *Controllers (C)* in the header. That is, the fields $A_1, \ldots, A_X$ represent the nodes of



(a) HELLO message.



(b) TOPOLOGY CONTROL message.

Figure 3.3: SD-OLSR packet formats.

Table 3.2: Description of the fields of HELLO and TOPOLOGYCONTROL packets.

	Field	Description
HELLO	ID	The type value of SDN resource.
	Reserved	Reserved for future use.
	HTime	HELLO message emission interval.
	Willingness	Willingness to act as a relay node for other nodes.
	Link Code	Information of the links with the neighbors or the status of the neighbors.
	Neighbor Interface Address	Address of the neighbor's interface.
TOPOLOGYCONTROL	Advertised Neighbor Sequence Number (ANSN)	Sequence number to keep track of the most recent status of the neighbor set.
	Advertised Neighbor Main Address (ANMA)	Address of the advertised neighbor.
	Lower Border	Specifies the lower limit of the range of selected IP addresses in ANMAs.
	Upper Border	Specifies the upper limit of the range of selected IP addresses in ANMAs.
	Ctrl+SW (X)	Number of nodes with controller+switch capability among the ANMAs.
	Controllers	Number of nodes with controller capability among the ANMAs.
	Reserved	Reserved for future use.

type-2, fields $A_{X+1}, \ldots, A_{X+C}$ constitute type-1, and fields $A_{X+C+1}, \ldots, A_N$ account for the switches (type-0). On receiving the Topology Control packets, switches extract the node information and generate the network topology. Further, switches filter out the controllers present in the network using the node-type information and compute the nearest controller as per the switch-initiated handoff described in Algorithm 3.2.

3.3.1 Operation of SD-OLSR Protocol

Figure 3.4 shows the snapshots of a software defined wireless mesh network involving mobile switches and controllers for four time instances. Each switch maintains a *candidate controller table* with available controllers arranged in their increasing order of cost to reach them. The switch always connects to the first controller in the table which incurs the minimum cost. At time $t = 1$, the network consists of two switches S1 and S2 and a controller C1. Since only a single controller is available, both switches connect to C1 for the flow-rules. At time $t = 2$, a new switch S3 and a controller C2 enter the network. The switches update their candidate controller table and switch S2 shifts to controller C2

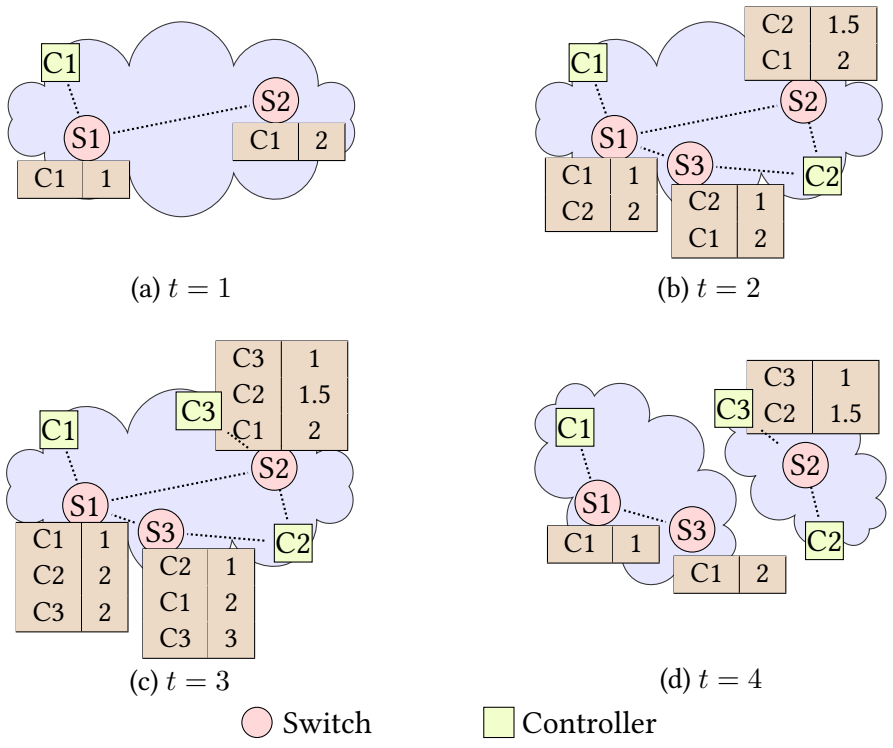


Figure 3.4: Snapshots of an SD-WMN running SD-OLSR protocol. The candidate controller table associated with each switch is provided near the switch.

for the flow-rules due to its close proximity compared to C1. The new switch S3 prefers its neighbor C2 to C1. The introduction of another controller C3 at time $t = 3$ does not affect switches S1 and S3, however, switch S2 changes to C3 from C2. At time $t = 4$, the network gets partitioned into two and the switches S1 and S3 get handed over to C1, the only controller in their partition. Even though switch S2 has the luxury of two controllers, it stays with C3, the nearest one in terms of ETX.

Software Defined OLSR protocol is the first of its kind in software defined wireless environments with the identification of SDN resources apart from offering the platform for in-band control as well as providing the global network view to controllers and switches. Even though SD-OLSR is designed for SIH, the protocol can be used in CIH to reduce the manual configuration effort required while adding new switches and controllers. The periodic HELLO and TOPOLOGY CONTROL messages keep the switches updated with the network dynamics and help to self-configure themselves, as shown in Sections 3.4 and 3.5. Table 3.3 provides a summary of the methods adopted for disruption tolerance in the proposed self-configuration schemes.

Table 3.3: Summary of disruption tolerance in self-configuration schemes.

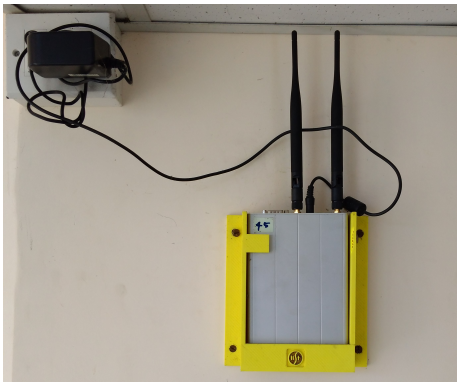
	CIH	**SIH**
Handoff technique	Controller role-change	Selection of new controller
Controller failure	• OLSR assigns infinite cost toward the failed controller. • Changes the role of nearest controller to *primary* if required.	• SD-OLSR assigns infinite cost toward the failed controller. • Connects to the nearest controller if required.
Switch failure	Uses alternate paths to route data packets.	Uses alternate paths to route data packets.
Link failure	• Uses alternate paths to route control and data packets. • Controller handoffs if required.	• Uses alternate paths to route control and data packets. • Controller handoffs if required.
Network partitioning	Uses multiple physical controllers.	Uses multiple physical controllers.

3.4 Experimental Setup for Self-Configuration Schemes

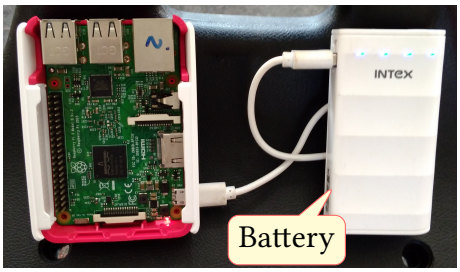
In order to analyze the efficacy of the proposed self-configuration scheme, we develop a software defined wireless mesh network environment involving mobile switches and

controllers. The testbed consists of five switches (two fixed and three mobile) and three controllers (one fixed and two mobile). Fixed switches are mounted on the wall (see Figure 3.5(a)) while the mobile switches and mobile controllers are attached to mobile platforms as shown in Figure 3.5(c). For the stationary switches, we use Alix3d3 single board computers whereas the mobile switches are realized using the Raspberry Pi computing boards. On the other hand, Lenovo ThinkCenter desktop is made to operate the stationary controller while Dell Vostro laptops are used as the mobile controllers. Switches and controllers run OpenFlow v1.3 [111] as the southbound interface over the OLSR and SD-OLSR protocols in CIH and SIH, respectively. We use Ryu v4.7 [112] as the network operating system for the control plane. The switches and controllers communicate via multihop path using in-band control. The detailed specifications of hardware and software used in the testbed are provided in Table 3.4.

In order to measure the performance of our self-configuration scheme in dynamic wireless environments, we designed the mobility of nodes (switches and controllers) in a such way to capture all possible behaviors in a wireless mesh environment including



(a) Fixed switch mounted on the wall.



(b) Mobile switch on the rolling platform.

(c) Fixed switch, mobile switch, and mobile controller.

Figure 3.5: Snapshots of SD-WMN testbed with switches and controllers.

Table 3.4: Hardware and software specifications of SD-WMN self-configuration testbed.

		Stationary	Mobile
Controller	Hardware	Lenovo ThinkCenter Intel Core i5@3.2 GHz, 4 GB RAM	Dell Vostro Laptop Intel Core i3@2.4 GHz, 2 GB RAM
	Operating System	Ubuntu 16.04 LTS	Ubuntu 14.04 LTS
	Routing Protocol	OLSR v0.6.5.4	OLSR v0.6.5.4
	SDN Controller	Ryu v4.7	Ryu v4.7
	Southbound Interface	OpenFlow v1.3	OpenFlow v1.3
Switch	Hardware	Alix3d3 AMD Geode LX800@500 MHz 256 MB RAM, Wistron DNMA92 802.11 a/b/g/n miniPCI WiFi module	Raspberry Pi 3 Model B, BCM2837 Quad Core CPU@1.2 GHz (64 bit) 1 GB RAM
	Operating System	Voyage Linux v0.9.2	Raspbian Jessie
	Routing Protocol	OLSR v0.6.5.4	OLSR v0.6.5.4
	Switch	Open vSwitch v2.3.1	Open vSwitch v2.3.0
	Southbound Interface	OpenFlow v1.3	OpenFlow v1.3

(*i*) mobility of switches, (*ii*) mobility of controllers, (*iii*) introduction of new controller, and (*iv*) controller failure. Each experiment is of 15 minutes duration and the mobility pattern followed by the nodes is classified into eight cases as shown in Figure 3.6 to include dynamic network behaviors. Three positions P1, P2, and P3 are marked in the layout as reference locations for the mobility of nodes.

Case 1: Case 1 represents the initial phase of the experiment. The network consists of five switches and a stationary controller C1. Since controller C1 is the only option, all switches connect to C1 for the flow-rules.

Case 2: In Case 2, a new controller C2 is introduced in the network near to position P1, which may result in handoffs in the neighborhood switches.

Case 3: Similar to Case 2, another controller C3 is introduced near position P3. Here, the network consists of all eight nodes.

Case 4: We introduce the mobility of switches in this stage with switch S1 moves from position P1 to P2, switch S2 shifts its position from P2 to P3, and switch S3 from P3 to P1.

Case 5: In order to analyze the network resiliency during a controller failure, controller C2 is disconnected from the network during Case 5. Therefore, the switches rely on C2 get handed over to either C1 or C3.

Case 6: Controller mobility is introduced in Case 6 by moving controller C2 from position P3 to P1 and switches get an opportunity to change their controller.

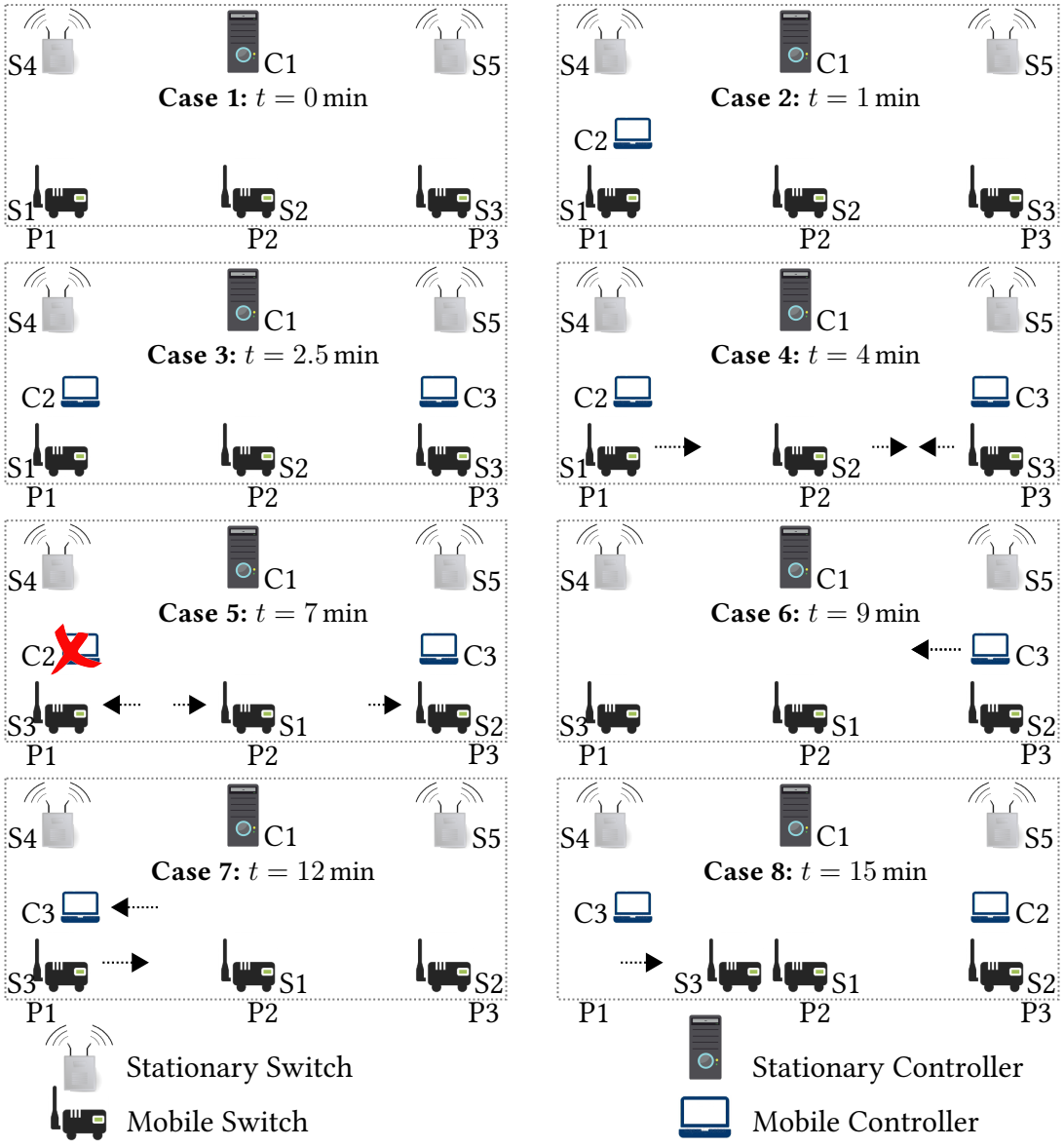


Figure 3.6: Experiment sequence involving mobility of switches and controllers.

Case 7: Switch S3 moves from position P1 to P2 and controller C2 rejoins the network near position P3. The switches near position P3 can exploit C2 by changing their controller to C2.

Case 8: Case 8 represents the final positions of nodes in the network.

Even though the nodes repeat the same pattern of physical movement as shown in Figure 3.6, the network topology may not follow the same sequence due to the uncertainty of wireless medium. Therefore, we repeated the experiment 10 times and used the samples

for further analysis.

3.4.1 Data Traffic Pattern

Since the primary objective of the testbed is to analyze the self-configuration of switches and controllers, it is required to generate sufficient SDN control packets, especially PacketIn and FlowMod messages, in the network. Therefore, we generate data traffic between all pairs of nodes using the *ping* utility [113]. That is, each switch in the network generates Internet Control Message Protocol (ICMP) packets to all other switches at a rate of 4 packets/second. The network control logic, as described in Algorithm 3.3, is realized as an SDN application running on the top of the controller. First, the algorithm computes the shortest path between the data packet's (the data packet is encapsulated within the PacketIn message) source and destination (Line 1). In case the switch that generates the PacketIn message is involved in the shortest path, the packet needs to be forwarded (Lines 3 and 4). Otherwise, the packet should be dropped to remove the redundant packets (Lines 5 and 6). Algorithm 3.3 returns the flow-rule as a FlowMod message with the $Match$ fields (i.e., the IP source and destination of the packet) generated in Line 2 and $Action$ in Line 8.

Here, the frequent changes in network topology may result in corresponding changes in the shortest path between any pair of nodes. That is, the flow-rules become invalid on a topology change which necessitates the need to update the flow-rules at the switches. Therefore, we set 60 seconds as the flow-rules' expiration time using the HardTimeout attribute, beyond which the packets in the flow result in flow-rule miss and the switches

Algorithm 3.3: Procedure of SDN control application.

Data:
S_{in} — Switch from where the PacketIn is received
Pkt — Data packet contained within the PacketIn message
ShortestPath(a, b) — Function returning the shortest path in terms of ETX between nodes a and b.

1 $P \leftarrow$ ShortestPath$(Pkt.src, Pkt.dst)$
2 $Match \leftarrow (IP.src = Pkt.src, IP.dst = Pkt.dst)$
3 **if** $S_{in} \in P$ **then**
4 $\quad Action \leftarrow$ FLOOD
5 **else**
6 $\quad Action \leftarrow$ DROP
7 **end**
8 **return** FlowMod$(Match, Action)$

request for new flow-rules. Since OLSR is used as the platform for in-band control, the control packets such as HELLO and TOPOLOGY CONTROL are handled using hidden flow-rules. Two benchmark schemes, (*i*) *No Handoff* and (*ii*) *No Handoff with Controller Failure*, are defined based on the existing multiple controller environments to compare the performance of the proposed schemes.

1. **No Handoff (NH):** In NH, the switches follow the mobility pattern shown in Figure 3.6 while the controllers are made stationary. Also, all controllers are made available to the switches for the entire duration of the experiment.

2. **No Handoff with Controller Failure (NH-CF):** Here, both switches and controllers follow the mobility model specified in Figure 3.6 along with the controller failure as shown in Case 5. Also, the switches are connected with all available controllers during the experiment.

The experiments are conducted as per the mobility sequence shown in Figure 3.6 using NH, NH-CF, CIH, and SIH under in-band control. The schemes NH, NH-CF, and CIH use OLSR for in-band control while SIH uses the SD-OLSR protocol for the process of self-configuration.

3.5 Performance Analysis

The primary objective of the self-configuration scheme is to adjust the network depending on the changing network characteristics in such a way that *the switches can request and obtain the flow-rules within minimum time and with minimum network control overhead.* Therefore, to quantitatively analyze the performance of the self-configuration schemes, we used the following seven metrics: (*i*) *OpenFlow overhead*, (*ii*) *OLSR overhead*, (*iii*) *PACKET-IN-FLOWMOD delay*, (*iv*) *FLOWMOD-PACKETIN ratio*, (*v*) *number of handoffs*, (*vi*) *controller-handoff time*, and (*vii*) *controller load.*

3.5.1 OpenFlow Overhead

Overhead computes the amount of additional information required to communicate the end-user data. As far as SDN is concerned, the overhead incurs primarily due to the traffic in SDN control channel, i.e., the OpenFlow messages. In our case, we divide the overhead into two types, (*i*) aggregate OpenFlow overhead and (*ii*) essential OpenFlow overhead, to distinguish the data-traffic-dependent overhead from the data-traffic-independent one.

3.5.1.1 Aggregate OpenFlow Overhead

Aggregate OpenFlow overhead is defined as the ratio of the total amount of OpenFlow messages (in bytes) to the total size of messages (in bytes) communicated over the network. That is, aggregate overhead computes the proportion of SDN control packets for carrying out the data traffic specified in Section 3.4.1. Figure 3.7 shows the overhead incurred by SDN control packets in both directions, i.e., controller-to-switch (C⟶S) and switch-to-controller (S⟶C). The SDN control messages initiated from switches and controllers are listed in Table 3.1.

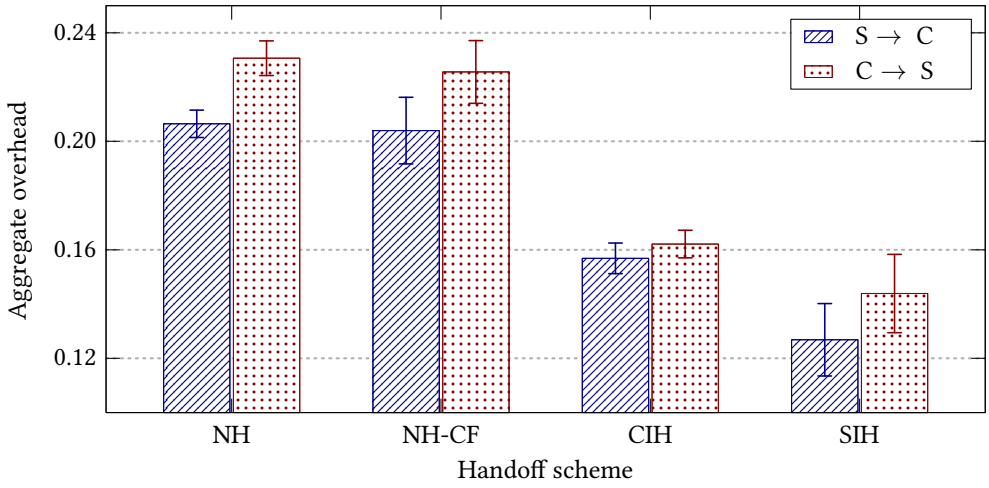


Figure 3.7: Aggregate OpenFlow overhead.

High overheads in NH and NH-CF are due to the switches' connection to all available controllers, i.e., upon a flow-rule miss, switches request flow-rules from all available controllers, thereby, increasing the overhead. Among the two proposed schemes, SIH performs better compared to CIH. More overhead in CIH is the result of periodic role-update (RoleRequest and RoleReply) messages, besides the traffic dependent messages, sent by the controllers to update their role toward each switch. Since switches are connected to all controllers in CIH, such role-update messages are essential to restrict the switches to request the flow-rules only from a single controller, thereby, minimizing the controller overhead, i.e., to change the switches from *equal* to *primary/secondary* role. In addition, it is required to change the role toward newly joined or re-entrant switches from default *equal* role to *primary* or *secondary*. SIH gets rid of the role-update messages by connecting the switches to only one controller at a time. It is clear from Table 3.1 that the response messages are of larger size compared to their request counterparts. Also, the frequent

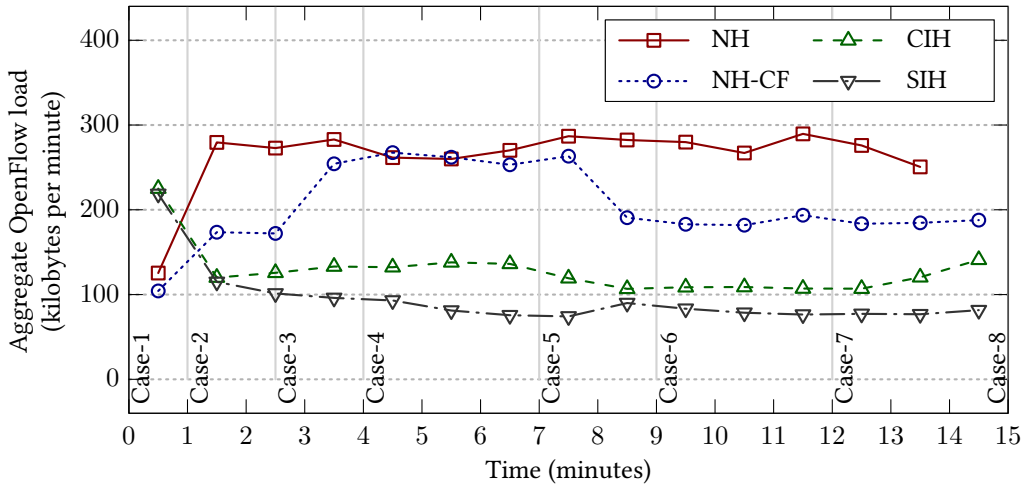


Figure 3.8: Aggregate OpenFlow load.

PacketIn-FlowMod messages resulting from the low HardTimeout of flow-rules make high controller-to-switch overhead.

The aggregate load incurred on the network due to the entire SDN control packets during the timeline of the experiment is shown in Figure 3.8. High overheads in the benchmark schemes NH and NH-CF are due to the redundant OpenFlow messages in the control channel. The proposed schemes SIH and CIH remove such a redundancy by restricting the flow-rule requests only to a single controller.

3.5.1.2 Essential OpenFlow Overhead

Essential overhead is defined as the ratio of the total size of *data-traffic-independent* messages (measured in bytes) to the total size of messages communicated over the network. The major proportion of aggregate overhead is due to the PacketIn and FlowMod messages generated from the flow-rule misses and the short expiration time (or HardTimeout) of flow-rules in wireless environments. Also, the flow-rules need to get updated depending on the frequent topology changes. However, only a subset of messages (FeaturesRequest, FeaturesReply, RoleRequest, RoleReply, MultipartRequest, MultipartReply, Hello, and Echo) are responsible for maintaining the effective switch-controller communication channel.

Figure 3.9 shows the overhead incurred due to the data-traffic-independent control messages. It can be observed that SIH incurs the least overhead in maintaining the control channel compared to all other schemes. On the other hand, CIH incurs the highest overhead of 15%, even significantly higher than the benchmark schemes. Such a high

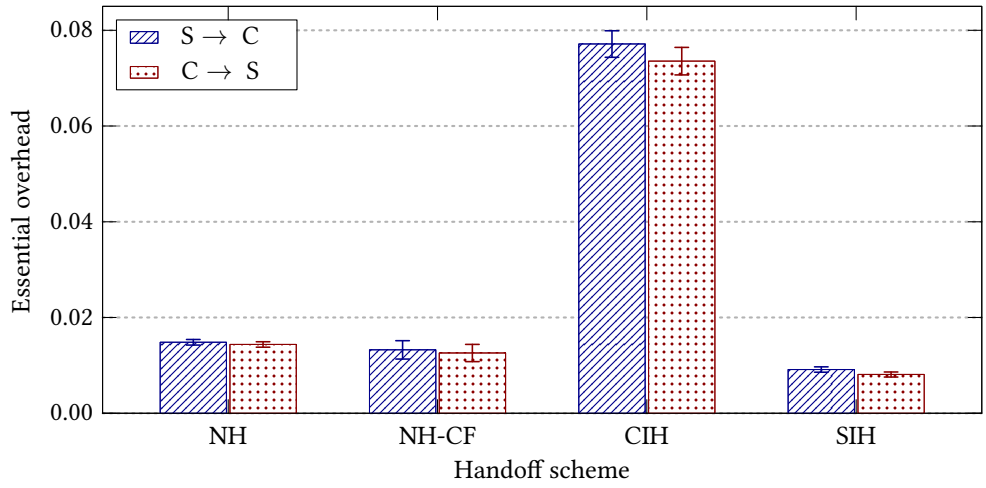


Figure 3.9: Essential OpenFlow overhead.

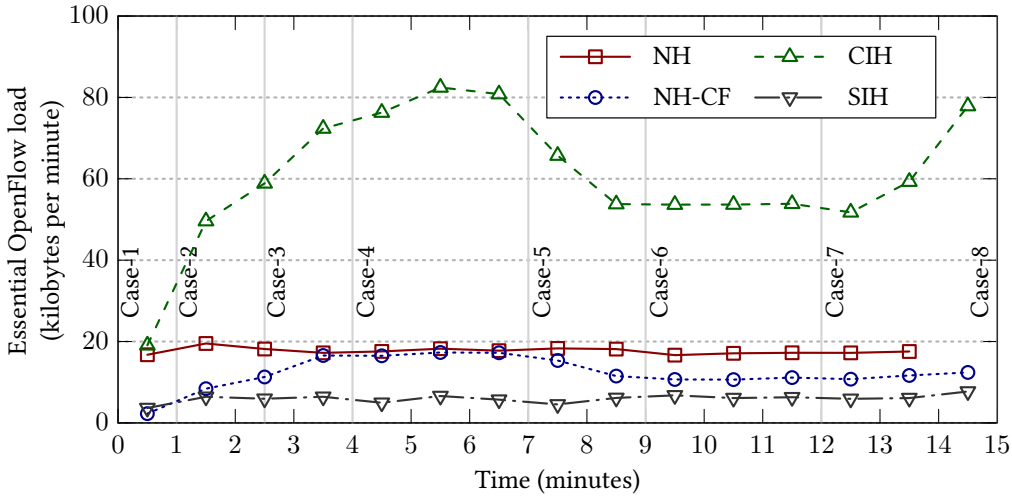


Figure 3.10: Essential OpenFlow load.

overhead in CIH is the result of periodic role-update messages (RoleRequest from controllers and corresponding RoleReply from switches) to compute the effective controller handoff. In fact, the overhead in CIH can be minimized by reducing the role-update interval, which is a function of the mobility pattern of nodes, at the controller. For example, in an SD-WN with stationary nodes, the update interval can be made significantly long in order to reduce the overhead. Due to the similar size of request-response messages that constitute the essential overhead, we did not observe a significant difference in overhead in terms of the direction of the control message. Figure 3.10 shows the timeline interpretation of essential OpenFlow load incurred in our experiments in eight cases concerned.

High load in CIH indicates its non-suitability toward highly mobile wireless environments with unstable links. It is important here to note that the essential overhead and load represent only a significantly small proportion of aggregate overhead and load shown in Figure 3.7 and Figure 3.8, respectively.

3.5.2 OLSR Overhead

The major hurdle in any software defined wireless network is in maintaining the control channel irrespective of the dynamic network behavior. The problem becomes more complex in mobile environments where traditional out-of-band approach is infeasible. Therefore, we use in-band control with OLSR protocol as the backbone for switch-controller communication in CIH, while the proposed SD-OLSR enabling SDN resource discovery is employed as the control channel in SIH. In order to measure the additional cost incurred for maintaining the control channel, we define OLSR overhead as the ratio of the total size of OLSR messages (measured in bytes) to the total size of messages (in bytes) communicated over the network. Figure 3.11 shows the OLSR overhead for the benchmark and the proposed handoff schemes. The benchmark schemes NH and NH-CF incur overhead of 10%. High overhead in SIH is due to the additional SDN-specific header fields added in the SD-OLSR protocol (see Figure 3.3) for automated resource identification. Figure 3.12 traces the behavior of OLSR load on the network during the experiment timeline. Except NH, all schemes follow the same pattern of OLSR load. Since NH does not involve any controller failure, the load is observed to be high throughout the experiment.

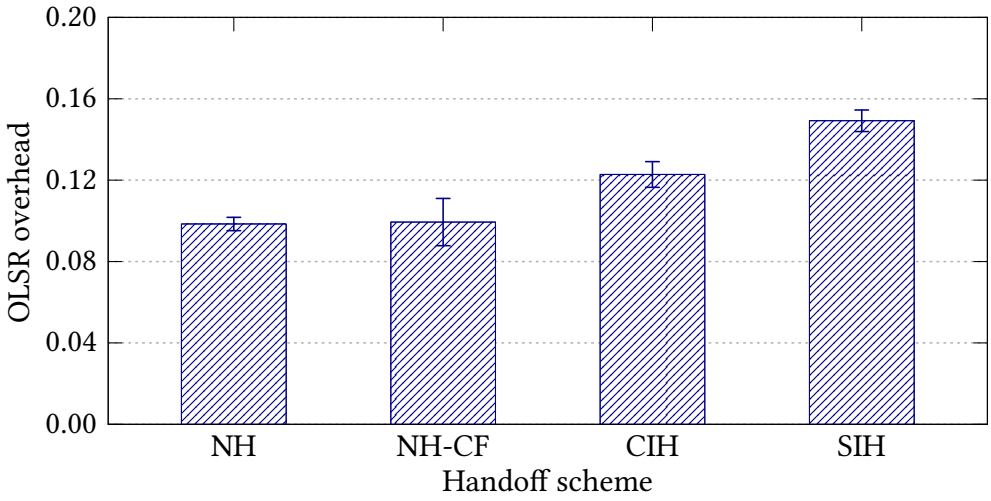


Figure 3.11: OLSR overhead.

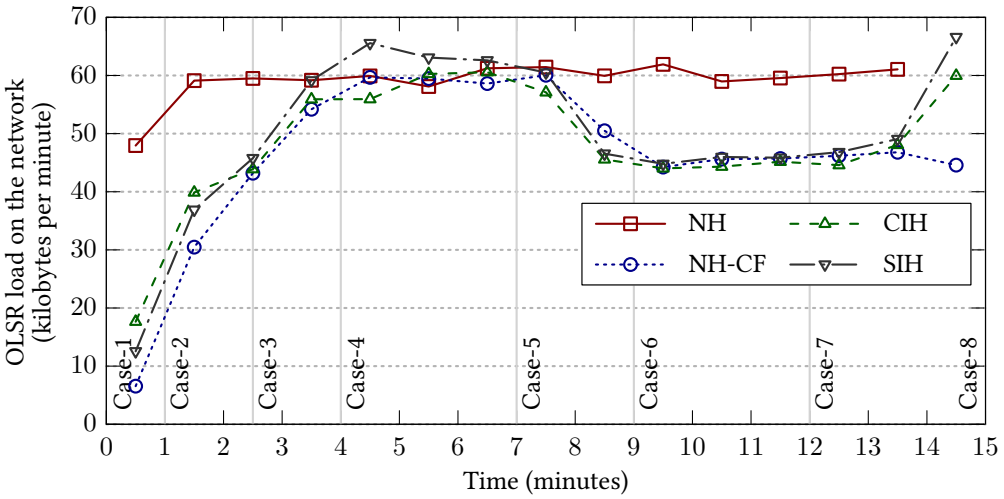


Figure 3.12: OLSR load.

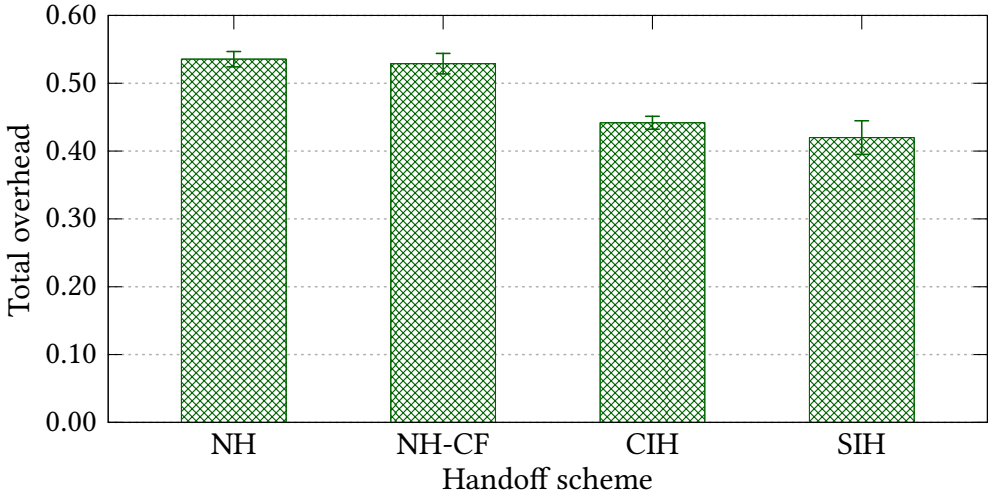


Figure 3.13: Total overhead (OpenFlow + OLSR) incurred by the handoff schemes.

In terms of the total overhead, i.e., the sum of aggregate OpenFlow overhead and OLSR overhead, Figure 3.13 reveals that both the benchmark schemes incur control overhead beyond 50%. On the other hand, CIH reduces the overhead to 44.17% even with the significant number of role-update messages. Similar to the behavior shown in Figures 3.7 and 3.9, SIH outperforms the other schemes with a control overhead of 41.99%. It is important for any network to minimize the control overhead to achieve maximum utilization of the communication channel for sending the user/application data. A major improvement can be achieved by realizing the network control in a maximum proactive way, i.e., controllers can send the required FlowMod messages without waiting for the

corresponding PacketIn requests. However, the controller can act in a reactive manner during inevitable circumstances such as the occurrence of unexpected network flows.

3.5.3 PacketIn-FlowMod Delay

The primary objective of the self-configuration schemes is to minimize the delay incurred for switches in communicating with the control plane (which is realized using multiple physical controllers), i.e., the switches need to receive the required flow-rules within minimum time on a flow-rule miss. Unlike out-of-band control, where the delay becomes minimum due to the dedicated control channel, in-band control faces data-traffic-dependent challenges due to the shared medium. We use PacketIn-FlowMod delay, i.e., the time it takes for a switch to receive a FlowMod message in response to a PacketIn request, to measure the switch-controller delay.

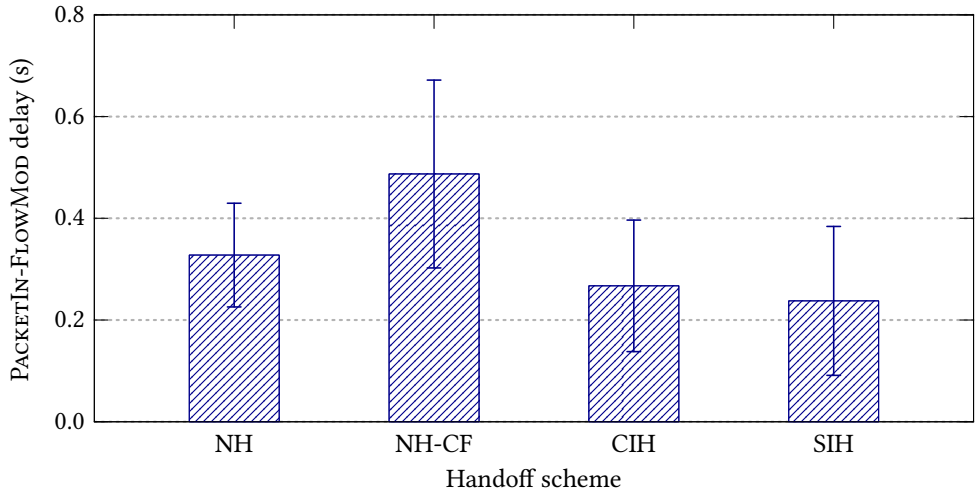


Figure 3.14: PacketIn-FlowMod delay.

Figure 3.14 shows the delay incurred by switches in receiving the flow-rules. We can observe that SIH offers minimum delay compared to CIH as well as the benchmark schemes. High delays in NH and NH-CF are due to their redundant PacketIn requests to all available controllers (even to the farthest controllers) in the network. In NH-CF, switches are forced to connect to the distant controllers during a controller failure (Case 5 in Figure 3.6), thereby, increasing the PacketIn-FlowMod delay.

3.5.4 FlowMod-PacketIn Ratio

Among the SDN control packets generated from the controller, only the FlowMod messages are responsible for modifying the state of a switch, i.e., a FlowMod message installs or modifies a flow-rule in the switch's flow-table and controls the corresponding network flow. A controller can work in either proactive or reactive manner to control the flows. In proactive approach, a controller can install the flow-rules in switches depending on the changing global network view rather than waiting for an explicit flow-rule request from the switch. On the other hand, in reactive approach, the controller sends flow-rules only as responses to the PacketIn messages resulting from the flow-rule miss events at the switches. At times, controller may not send flow-rules to the switches as well. We use reactive approach for our experiments. FlowMod-PacketIn ratio measures the controller response to switch requests, i.e., the ratio of the number of FlowMod messages received to the number of PacketIn messages sent. Figure 3.15 shows the behavior of controller response in term of PacketIns. Since we use reactive approach in the benchmark as well as the proposed handoff schemes, they exhibit similar behavior with ratio beyond 0.8. Among the four schemes, NH and SIH offer the highest ratio of 0.86 while NH-CF performs the least with ratio of 0.82.

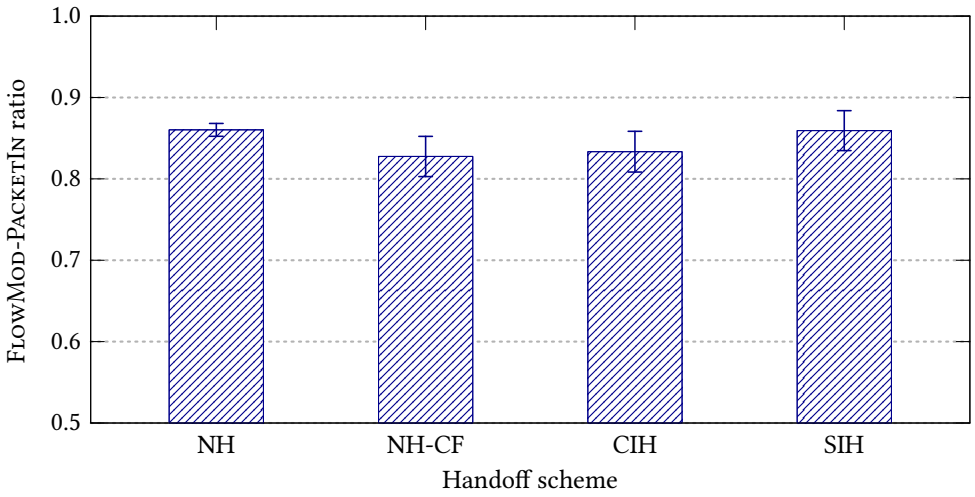


Figure 3.15: FlowMod-PacketIn ratio.

3.5.5 Number of Controller Handoffs

Even though the handoff schemes are designed with an objective of improving the network performance by connecting switches to the nearest controller, the schemes may have

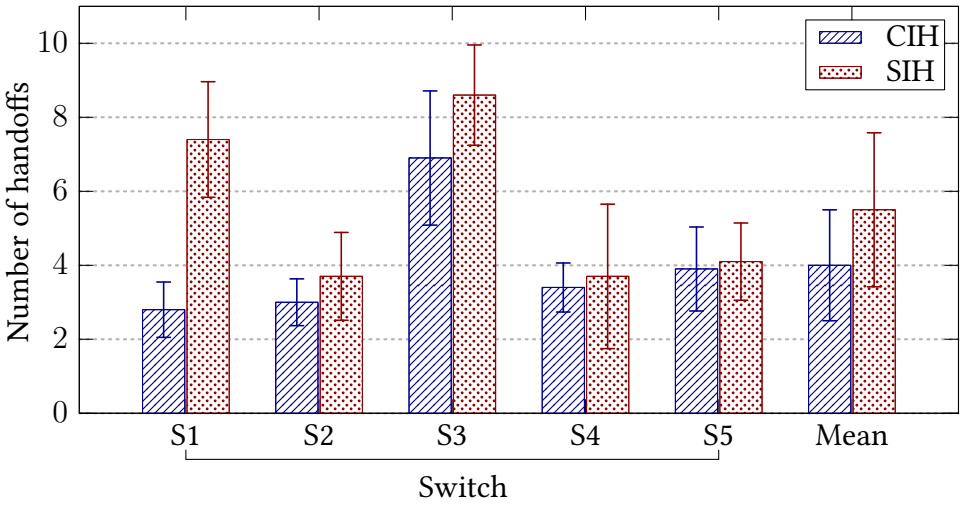


Figure 3.16: Number of controller handoffs.

a negative influence if the handoffs occur at unacceptably high frequencies or at inappropriate times. More handoffs result in significant number of control messages and even data packet loss. On the other hand, very less number of handoffs may reduce the performance of highly mobile environments. Therefore, it is important to have a limit on the number of handoffs based on the application. Figure 3.16 shows the number of handoffs occurred in CIH and SIH during our experiments, with x-axis being the switches involved in the experiment and y-axis, the number of handoffs. The mean number of controller handoffs is shown as a separate histogram. We exclude the benchmark schemes NH and NH-CF due to the absence of controller handoffs. It is clear from Figure 3.16 that the number of handoffs in SIH is higher than CIH in all switches involved. On an average, CIH and SIH provide 4 and 5.5 handoffs, respectively.

3.5.5.1 The Role of α and τ

As discussed above, it is important to limit the number of controller handoffs depending on the application and requirements. We provide two parameters, (*i*) the smoothing factor α and (*ii*) the handoff threshold τ, to make our self-configuration scheme suits to different application scenarios. Considering the example of software defined vehicular networks, it is difficult to maintain a switch residing within the vehicle under a single controller due to the high speed of the vehicle. Therefore, it is necessary to handover the vehicle-mounted switch to the upcoming controller at the earliest to maintain the uninterrupted control channel using a high value of α and a low value of τ. On the contrary, a low value of α

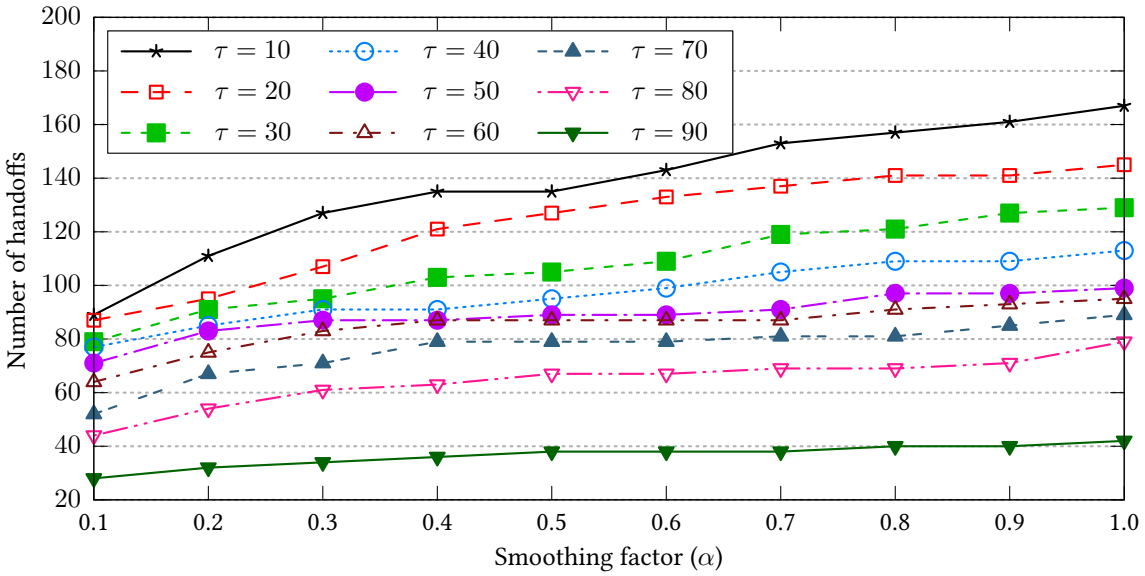


Figure 3.17: Total number of handoffs with varying values of α and τ.

along with a high value of τ may suffice for the low-speed mobility environments such as campus networks where pedestrians carry the mobile nodes. Figure 3.17 shows the number of controller handoffs with varying values of α and τ from an SD-WMN testbed involving six switches and two controllers where four switches follow random mobility pattern. As α increases, the exponential smoothing function prefers the instantaneous values of ETX more while computing the handoff decisions, thereby, resulting in more number of handoffs proportional to the number fluctuations in the handoff metric. However, more the value of alpha, less will be the number of handoffs since the switches shift to a new controller only if the cost difference is beyond the threshold. As shown in Figure 3.18, the time between handoffs reduces as the number of handoffs increases. In this context, our proposed controller handoff schemes offer sufficient flexibility to suit the next generation wireless networks, involving different node mobility patterns, in terms of real-time network management with self-configuration as envisioned in [114].

3.5.6 Controller Handoff Time

Controller handoff time is an important parameter influencing the uninterrupted network performance. The more time the handoff process takes for completion, the more will be the network interruption and packet loss. That is, long handoff time makes the switches wait for long for flow-rules, thereby, resulting in data packet loss. Since the handoff procedures are different, we define the metric handoff time separately for CIH and SIH. In CIH,

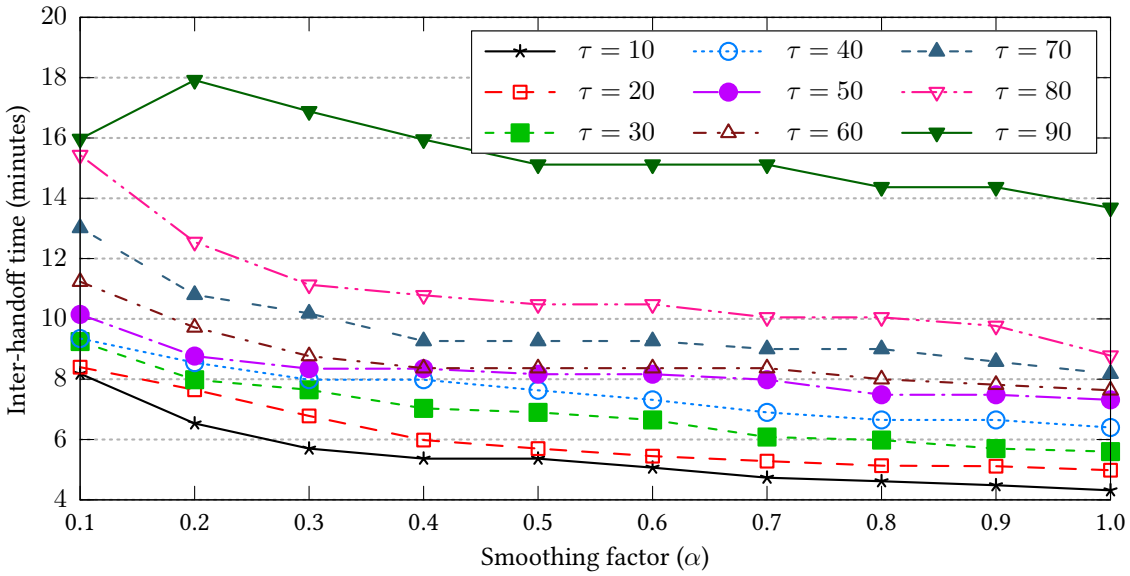


Figure 3.18: Inter-handoff time with varying values of α and τ.

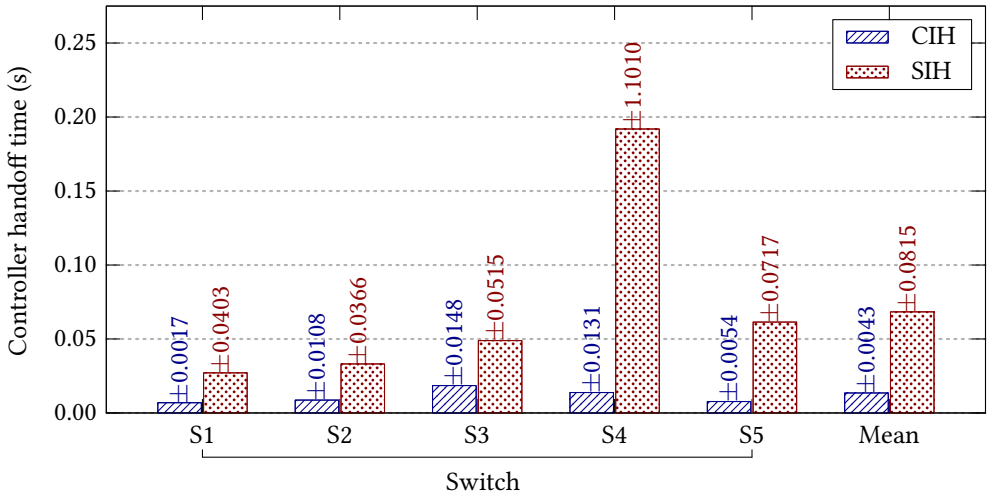


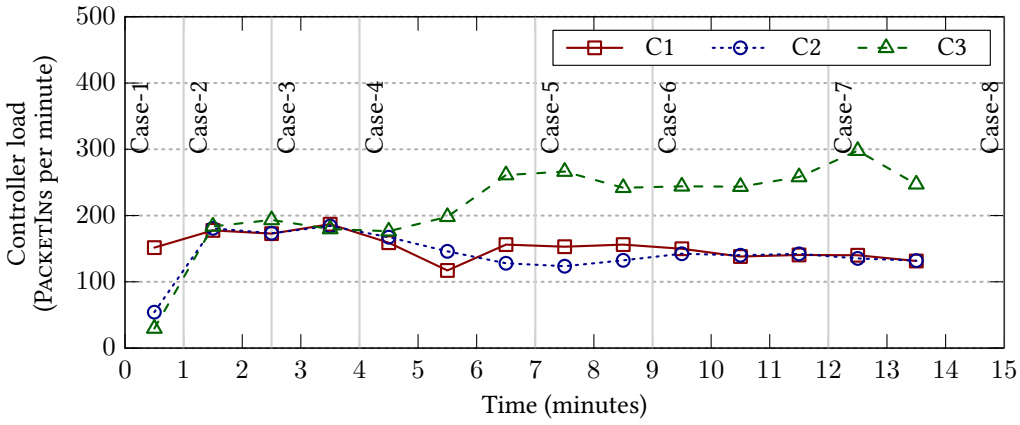
Figure 3.19: Controller handoff time. The standard deviation is provided on the top of each bar.

handoff time is defined as the time for a controller to receive a RoleReply message from a switch with the confirmation of a *primary* role upon a *primary* RoleRequest to that switch. On the other hand, handoff time for SIH is defined as the time for a switch to receive the first FlowMod message, with CONTROLLER as the action, for a flow-rule miss during a switch-controller connection setup. It is clear from Figure 3.19 that CIH incurs least controller handoff time compared to SIH. Due to the prerequisite for the switches to have an established connection with all controllers, CIH requires only a two-way hand-

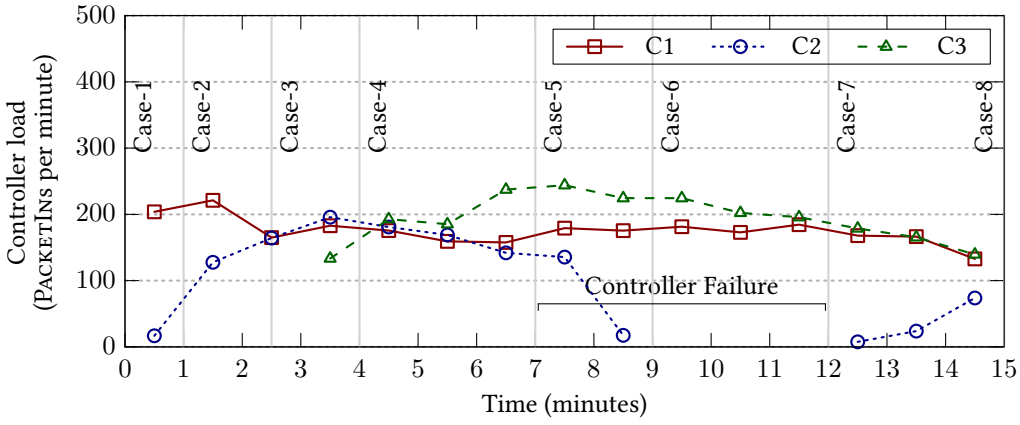
shake (a RoleRequest and RoleReply message) to complete the handoff process. On the contrary, in SIH, the switches follow the same procedure as that of a new connection setup while handing over to the new controller. That is, SIH involves Hello, FeaturesRequest, FeaturesReply, MultipartRequest, MultipartReply, and FlowMod messages to complete the handoff procedure. Unlike CIH, switches in SIH close their connection with the existing controller before handing over to the new controller, thereby, eliminating the requirement for a controller-controller interaction as used in [20].

3.5.7 Controller Load

Apart from ensuring resilience from single point-of-failure of controller in SDN, multiple controllers provide an added benefit of load balancing in terms of the number of Pack-

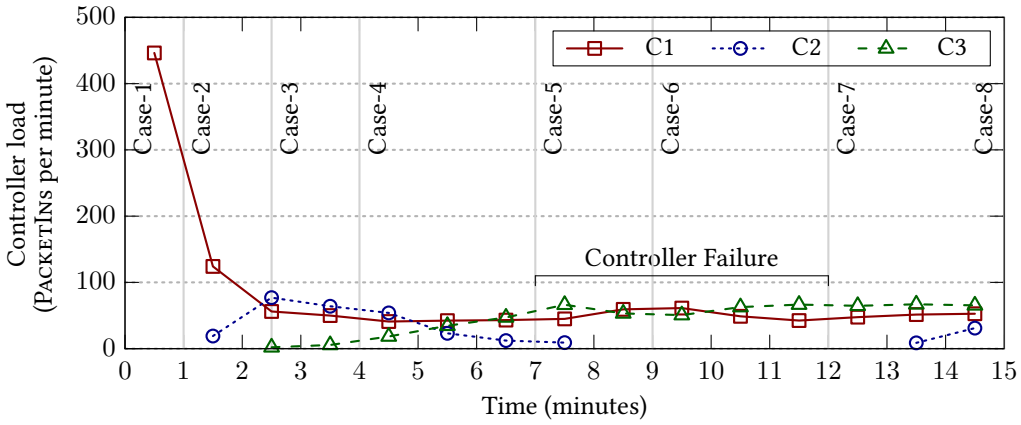


(a) No Handoff (NH).

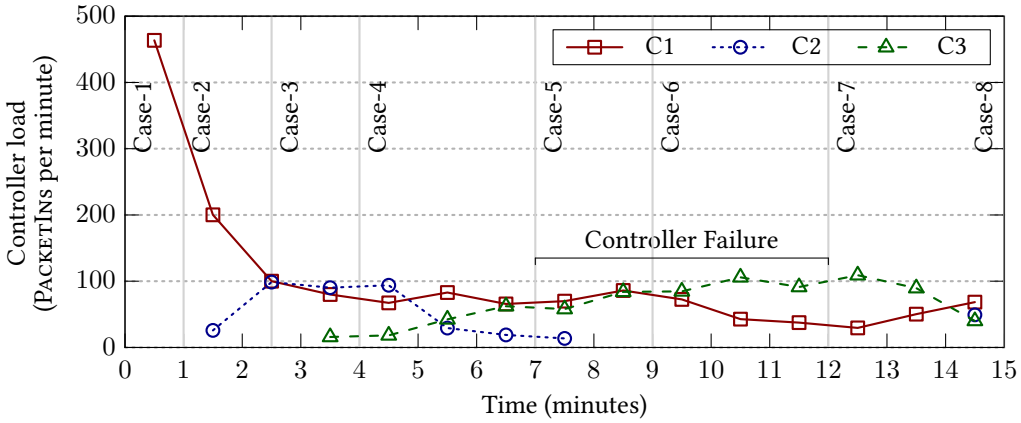


(b) No Handoff with Controller Failure (NH-CF).

Figure 3.20: Load incurred on the controllers in different handoff schemes. (Figure continued in the next page.)



(c) Controller-Initiated Handoff (CIH).



(d) Switch-Initiated Handoff (SIH).

Figure 3.20: Load incurred on the controllers in different handoff schemes.

etIn messages in the network. It is important to handle the controller load in networks involving large number of high data-rate flows. Since PacketIn messages represent the network control (flow-rule) request, we define controller load as the total number of PacketIn messages handled by a controller per unit time. Figure 3.20 shows the load incurred per minute on the three controllers in the benchmark as well as the proposed handoff schemes during the timeline of our experiments. Since each switch is connected to all available controllers in NH and NH-CF, the PacketIn requests are send to all controllers, thereby, increasing the controller load as visible in Figures 3.20(a) and 3.20(b). Better connectivity of switches toward controller C3 can be observed from the high controller load beyond time $t = 4$. On the failure of controller C2, its load gets shared across C1 and C3 during $8.5 \leq t \leq 12.5$ minutes. The equal sharing of controller load is more pronounced in CIH than SIH. The load gets re-adjusted among all controllers when controller C2 re-

joins the network at $t = 12.5\,\text{mins}$. Figures 3.20(c) and 3.20(d) show the inherent ability of our proposed schemes in providing self-configuration with a significantly reduced controller load.

3.6 Major Observations

The performance of our proposed self-configuration scheme from different dimensions proves the efficacy of our approach in handling link disruptions in the control channel of software defined wireless environments involving mobile nodes, *including switches and controllers*. Even though both the proposed schemes are promising candidates for controller handoffs, the results provide indications toward their applicability in different environments. Moreover, the proposed handoff schemes and SD-OLSR protocol are compatible with SD-WNs operating over the out-of-band control channel with nodes equipped with multiple NICs. In such cases, the control overhead incurs only on the dedicated channel whereas the switches can utilize the entire data channel for communicating user/application data, thereby, improving the throughput and end-to-end delay. In short, the major observations on two controller handoff schemes are as follows:

Controller-Initiated Handoff

1. CIH is best suited to wireless environments with limited mobility as well as link disruptions.
2. CIH restricts its scope toward environments where configuration of nodes is an easy process.
3. High essential overhead necessitates networks with high data-rate links for effective CIH.
4. CIH is suitable for time sensitive SD-WNs that require minimal interruptions from controller handoffs.

Switch-Initiated Handoff

1. SIH is suited for environments with high degree of node (including switch and controller) mobility and dynamic addition/removal of nodes.
2. SIH is applicable to environments where the real-time configuration of network nodes is difficult or infeasible.

3. SIH forms a promising candidate for fault-tolerant SDN systems where controllers and switches can be introduced with near-zero configuration effort.

4. Low essential overhead makes SIH suitable for networks with low data-rate links.

3.7 Summary

In this chapter, we focused on handling link disruptions in SDN control channel in software defined wireless networks involving switch and controller mobility. We first approached the problem of controller disconnection using multiple physical controller deployments with a novel controller handoff scheme Controller-Initiated Handoff. The prerequisite that the switches need to connect to all controllers and the requirement of explicit real-time configuration of switches and controllers make CIH difficult to get adopted in highly mobile wireless environments involving frequent link disruptions in the SDN control channel. Therefore, we moved forward to a Switch-Initiated Handoff mechanism with an extended Software Defined OLSR protocol capable of providing automated SDN resource discovery along with the controller handoff mechanism. The results provided in Section 3.5 showed the efficacy of the proposed self-configuration schemes in different wireless environments operating under in-band channel. As the next step, we design and develop a software defined disruption tolerant networking framework in Chapter 4 to resolve the issues emanating from link disruptions in data plane.

Chapter 4

Toward Software Defined Disruption Tolerant Networks

> "*The key is to embrace disruption and change early. Don't react to it decades later. You can't fight innovation.*"
>
> — Ryan Kavanaugh

One of the most important tasks of any SD-WN is to respond to the link disruptions in data plane at the earliest in order to meet the required QoS requirements. Link disruptions in data plane can occur due to the mobility of end-user devices as in WLANs (discussed in Section 1.3.1), mobility of switches as in WMNs (described in Section 1.3.2), or due to the environmental factors in fixed wireless backhauls (as revealed in Figure 1.8). In traditional networking systems, long-term disruptions are handled using Disruption Tolerant Networking (DTN) approach [81] while issues emanating from short-term disruptions are resolved via traffic re-routing or rate adaptation techniques [69]. In [115], we classified the disruptions into three: (*i*) short-term, (*ii*) medium-term, and (*iii*) long-term, as shown in Figure 4.1. Short-term disruptions are of the duration below 10 seconds while the medium-term disruptions span between 10 seconds and 6 minutes. The short and medium-term

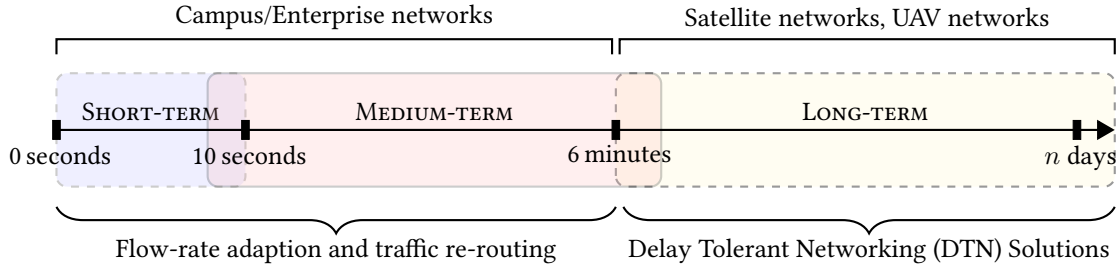


Figure 4.1: Classification of link disruptions based on disruption duration.

disruptions occur primarily in campus and enterprise networks. The disruptions of durations beyond 6 minutes are considered as long-term disruptions and are handled using traditional DTN techniques. However, such disruption-handling techniques integrated with the forwarding devices are not sufficient to achieve the requirements of next generation wireless networks that are characterized by extremely low end-to-end latency and dynamic user demands. In this chapter, we propose a Software Defined Disruption Tolerant Network (SD-DTN) to handle link disruptions using the existing TCP/IP protocol stack. In SD-DTN, we introduce the concept of controlled buffering of packets at the switches using a new STORE action by exploiting the storage space available within the switches. Using the STORE action, controllers can make the switches buffer packets during link disruptions in a network-flow's path and forward them as the path gets re-established. Such a buffering at the intermediate switches provides performance benefits in terms of end-to-end delay as well as energy consumption incurred in end-to-end packet re-transmissions resulting from packet losses. Therefore, we propose the SD-DTN framework with the following components:

1. A temporal graph-based network model for SD-DTNs along with a network prediction algorithm using Markov chains.

2. An Earliest Arrival Path with Minimal Storage Time (EAPMST) controller algorithm for estimating the temporal path for a flow, thereby, computing the buffer/forward control decision.

3. An SDN-controlled buffering mechanism for switches to store packets during link disruptions.

4. A new action STORE for SDN controllers to enable switches to buffer packets.

5. A new SD-DTN switch prototype and software switch capable of controlled buffering.

4.1 Network Model for SD-DTN

In order to design the SD-DTN framework, we model the network involving link disruptions using a graph theoretic approach. However, due to the frequent link disruptions in wireless networks, the network topology changes more often, thereby, making a single graph instance inadequate to characterize the entire network behavior. Therefore, we use

the concept of temporal graphs or time varying graphs [116,117] to represent the network dynamics by attributing a time dimension to the network nodes and links.

In our network model, the network behavior is captured using a sequence of static graphs $\mathcal{S} = \langle G_1, G_2, G_3, \ldots \rangle$, where the graph $G_t = (V_t, E_t)$ represents the network at time t, $t \in \mathbb{N}$. For each graph G_t in the sequence $\mathcal{S}$, V_t constitutes the set of network nodes[1] (or vertices) in the network and E_t represents the set of links (or edges) between the nodes at time t. We assume that the graph G_t does not change its state during the interval $[t, t+1)$ and the set of nodes V_t (*not the position of nodes*) remains the same for all t. Therefore, a network node entering or leaving the network can be considered as an isolated node in the graph at appropriate time interval. Since the set of nodes remains the same, the network dynamics can be portrayed using the changes in the behavior of links.

We make use of Markov chains to capture the state-changes of all possible links in the network. The behavior of each possible edge (u, v) of the graphs in the sequence is represented as a two-state discrete time Markov chain $\{X_{(u,v),1}, X_{(u,v),2}, X_{(u,v),3}, \ldots, X_{(u,v),t}\}$ with state space $\{0, 1\}$, where

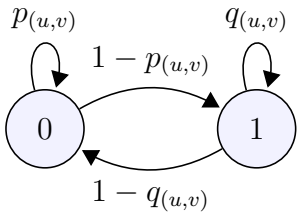
$$X_{(u,v),i} = \begin{cases} 1 & (u,v) \in E_i, \text{ (i.e., there exists the link } (u,v) \text{ in } G_i \text{ at time } t = i), \\ 0 & \text{otherwise (i.e., the link } (u,v) \text{ is absent in } G_i \text{ at time } t = i). \end{cases}$$

In other words, the states of all possible links at time i form the state of the network G_i. In networks that follow periodic connection patterns, $X_{(u,v),i} = X_{(u,v),i+k\Delta}$ for all $(u, v) \in E$, where Δ represents the periodicity interval and $k \in \mathbb{N}$. On the contrary, the future states of link (u, v) become unknown in aperiodic networks. However, in order to realize network control and compute the network control decision, it is necessary to estimate the future states of the network from the available history of network states.

Since we have the network sequence $\mathcal{S} = \langle G_1, G_2, G_3, \ldots, G_t \rangle$ at any time t, we need to predict the future graph sequence $\mathcal{S}'_t = \langle G'_{t+1}, G'_{t+2}, \ldots, G'_{t+M} \rangle$[2]. In order to estimate the future states of the network, we exploit the t-step transition probability of Markov chains. Figure 4.2 shows the state-transition diagram and the corresponding transition probability matrix $P_{(u,v)}$ of an edge (u, v) in the network sequence $\mathcal{S}$. In the figure, $p_{u,v}$ represents the probability that the edge (u, v) remains at state 0 (transition from state 0 to itself) and $q_{u,v}$ the probability that the edge remains at state 1 (transition from state 1 to itself). Given the state of an edge (u, v) at time t, the probability of that edge

[1]Nodes in a graph represent the switches and controllers in the network while the edges represent the wireless links between them.

[2]We denote the real network sample at any time t as G_t and the predicted network at time t as G'_t.



$$P_{(u,v)} = \begin{pmatrix} p_{(u,v)} & 1 - p_{(u,v)} \\ 1 - q_{(u,v)} & q_{(u,v)} \end{pmatrix}$$

Figure 4.2: State-transition diagram and the corresponding transition probability matrix of an edge (u, v).

being present at time $t + k$ is computed as

$$\mathbb{P}(X_{(u,v),t+k} = 1) = \left(Q^{\mathsf{T}}_{(u,v),t} P^{k}_{(u,v)}\right)_2, \tag{4.1}$$

where $Q_{(u,v),t}$ represents the column vector of the probability distribution of $X_{(u,v),t}$ at time t. The subscript 2 at the right hand side of Eqn. (4.1) denotes the second element of the resulting row vector.

For estimating the state of the network at time $t + 1$, we compute the probability distributions of all possible edges of the network at time $t + 1$ and add the edges with the estimated probabilities. The procedure for predicting graphs for the future M time instances is described in Algorithm 4.1. For each time instant $t + k$, the graph G'_{t+k} is initialized with the vertex set V containing all nodes in the network and an empty edge set E_k (Lines 3 and 4). For each possible edge (u, v), the probability p of that edge being present at time $t + k$ is computed using Eqn. (4.1) (Line 6). Further, the edge (u, v) is added to the edge set E_{t+k} with probability p. The graph is appended to the sequence $\mathcal{S}'_t$ upon adding the possible edges (Line 9). Since we limit the number of future time-steps to M and the maximum number of edges being $|E| = |V| \times (|V| - 1)$, the worst-case time complexity of Algorithm 4.1 is estimated to be $\mathcal{O}(M|V|^2)$.

It is important here to note that the proposed network model is independent of the mobility model of the nodes, i.e., Algorithm 4.1 remains the same for wireless environments irrespective of the node mobility pattern. In fact, Markov model is an important tool in modelling the link characteristics of different classes of wireless networks including cellular networks [118–121], WLANs [122–124], wireless mesh networks [125–129], and vehicular networks [130–134]. In our model, the transition probability matrix $P_{(u,v)}$ can be computed either from the prior knowledge of the past network states or from the network-state samples collected in real-time. In the former case, the probability transition matrices can be computed beforehand and can be used for computing the probability dis-

Algorithm 4.1: Procedure for predicting graphs in the sequence $\mathcal{S}'_t$.

Data:
V — Set of network nodes/vertices
$G_k = (V_k, E_k)$ — Graph at time k and $V_k = V$ for all k

$\mathcal{S}'_t \leftarrow \langle\rangle$
for $k = 1 : M$ **do**
  $G'_{t+k} \leftarrow (V, E_{t+k})$
  $E_{t+k} \leftarrow \{\}$
  for each (u, v) where $u, v \in V$ and $u \neq v$ **do**
    $p \leftarrow \left(Q^{\mathsf{T}}_{(u,v),t} P^{k}_{(u,v)}\right)_2$
    Add (u, v) to E_{t+k} with probability p
  end
  Append G'_{t+k} to the sequence $\mathcal{S}'_t$
end
return $\mathcal{S}'_t$

tribution of edges in Algorithm 4.1. However, in the latter case, i.e., in the absence of prior knowledge of the network states, the values of $P_{(u,v)}$ can be computed in real-time and need to be updated with each recent sample to make the matrix more accurate. In both cases, the values of $Q_{(u,v),t}$ can be extracted only from the real-time network samples. For our experiments, we compute the transition probability matrices beforehand using the past network samples collected from the experiments involving stationary, group, and random mobility models so that the controller is relieved from the computational overhead in calculating $P_{(u,v)}$ in real-time.

4.2 Earliest Arrival Path with Minimal Storage Time Algorithm for SD-DTNs

Even though we modelled and predicted the future network states in Section 4.1, the network control decision, i.e., STORE or FORWARD, depends on the *possibility of establishing a path toward the destination in future*. Whenever a packet reaches a switch in SD-DTN, the switch can forward the packet if there exists a link that forms a path toward the destination. In the absence of such a link, the switch needs to buffer the packet until the establishment of a suitable link. Therefore, the fundamental problem of any SD-DTN is *when*, *where*, and *till what time* the packets corresponding to a flow need to be buffered.

Unlike traditional data forwarding devices, SDN switches use advanced memory tech-

nique called Ternary Content Addressable Memory (TCAM) for maintaining flow-tables to achieve maximum efficiency in comparing the packet attributes with the flow-rules [111]. TCAM is also equipped with a *don't care (or wildcard)* option for searching content in the flow-table by discarding unwanted match-fields while comparing with the packets' attributes. However, TCAMs are characterized by high cost and limited memory size which restrict their use only to storing the flow-tables. That is, exploiting TCAM for buffering packets leaves little room for the flow-rules, thereby, compromising the objectives of flexible network control in SDN. Therefore, TCAMs or primary memory can be considered only for buffering short packets of negligible storage times while packets of large size or significant buffering time can be buffered in the secondary memory.

Since the memory space available at the switches forms an influential factor in the controlled buffering process, SD-DTN necessitates a route and buffer control algorithm minimizing the packets' buffering time at the intermediate switches in a flow's path. Therefore, we design an *Earliest Arrival Path with Minimal Storage Time* (EAPMST) buffering and routing algorithm for the SDN controller in order to estimate a possible *temporal path* minimizing the buffering time while ensuring the earliest delivery of packets. EAPMST makes use of the predicted graph sequence $\mathcal{S}'_t$ from Algorithm 4.1 and computes a temporal path toward the destination from the switch where the packets reside, i.e., the switch that sends the flow-rule request (i.e., PacketIn message) to the controller.

Unlike static graphs, where the shortest path is of prime importance, temporal graphs are characterized by different kinds of paths such as shortest path (in terms of the distance), fastest path, latest departure path, and foremost path or earliest arrival path [135, 136]. Among the four, we consider *foremost* or *earliest arrival* path for EAPMST due to the significance of DTNs in delivering the packets at the earliest possible time. Besides earliest arrival path, the buffering time is an important concern in SD-DTNs due to its influence in making the buffer space available for packets of the upcoming flows. Therefore, we define the SD-DTN route and buffer control problem as follows: *Given a graph sequence* $\mathcal{S}'_t = \langle G'_{t+1}, G'_{t+2}, \ldots, G'_{t+M} \rangle$*, the current node* v_0*, and the destination node* v_d*, find an earliest arrival path*

$$\mathcal{P} = \langle\!\langle\, v_0(w_0), v_1(w_1), v_2(w_2), \ldots, v_d \,\rangle\!\rangle, \tag{4.2}$$

such that $\sum_{v_i \in \mathcal{P}} w_i$ *is minimum.* In Eqn. (4.2), $v_i \in V$ and w_i represents the flow's buffering or waiting time at vertex v_i for the link (v_i, v_{i+1}) to get established. In the input sequence $\mathcal{S}'_t$, the link (v_i, v_{i+1}) gets established at time $t + k$, i.e., $(v_i, v_{i+1}) \in E_{t+k}$, where

$k = i + \sum_{j=0}^{i} w_j$, the sum of link delays and cumulative waiting times at nodes from v_0 to v_i. The procedure for generating the temporal path $\mathcal{P}$ ensuring earliest delivery and minimum buffering time at the intermediate nodes is described in Algorithm 4.2.

In Algorithm 4.2, each node v in the network is associated with a triple (h_v, p_v, c_v), where h_v represents the minimum distance (or the number of hops) from source node v_0 to node v, p_v is the predecessor node of v in the earliest arrival path, and c_v is the cumulative buffering time at the nodes in the earliest arrival path from v_0 to v. In Lines 1–4 the triple associated with each node is initialized to appropriate values and marks source node v_0 as visited. Further, the algorithm starts visiting the unvisited neighbors of visited nodes (Lines 7–14). While visiting a neighbor u of a visited node v, the triple (h_u, p_u, c_u) is updated only if the cumulative buffering time for reaching u via v is less than the cumulative buffering time through the existing predecessor node p_u (Lines 9–11). In addition, the waiting times of visited nodes are incremented by 1 (Lines 15 and 16) in order to explore the path toward destination due to the temporal nature of graphs, i.e., the destination can come in contact with the already visited nodes in future. Further, the neighbors of v are marked as visited by including them in the set $\mathbb{V}$ (Line 18). The exploration phase stops at the moment when the algorithm visits the destination node v_d in order to ensure the earliest arrival of packets, which is one of the primary objectives of any DTN. In the absence of a temporal path, the algorithm exhausts all M predicted graphs in the sequence $\mathcal{S}'_t$ and stops the exploration phase.

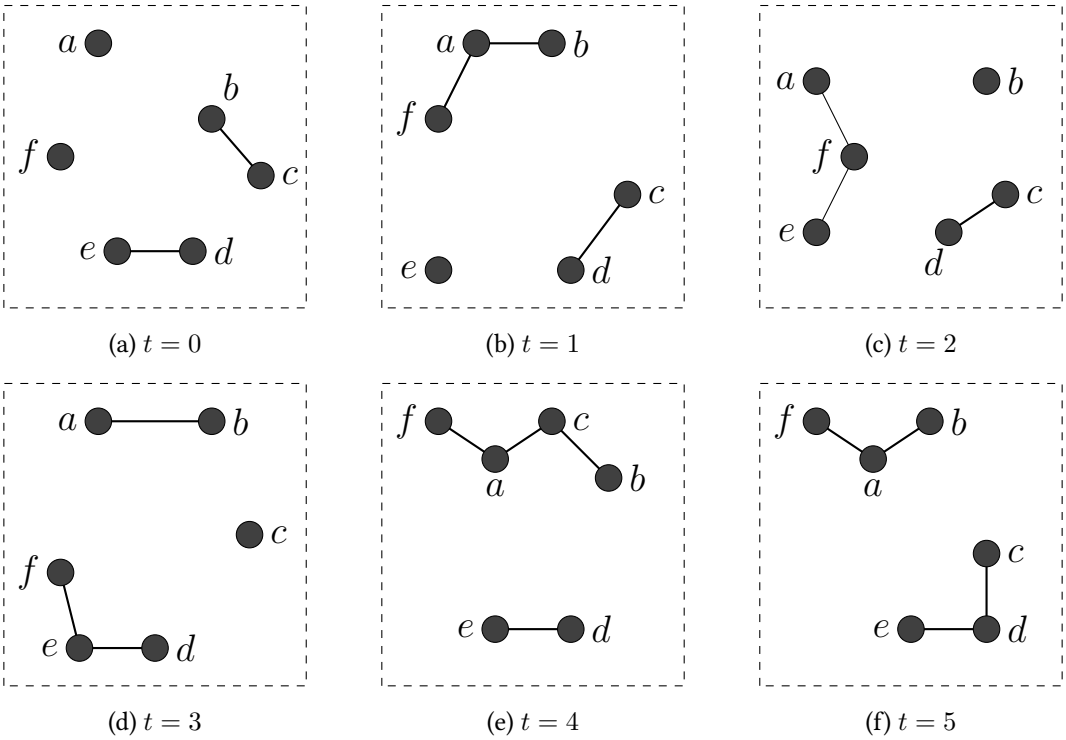
In case the destination node v_d is reached, i.e., $v_d \in \mathbb{V}$, the algorithm traces the route $\mathcal{P}'$ toward source node v_0 from v_d using the predecessor value p_v associated with each node (Lines 21–26). Further, the temporal path $\mathcal{P}$ is generated by populating the route $\mathcal{P}'$ with the buffering time required at each node which can be computed by traversing the predicted graph sequence $\mathcal{S}'_t$ (Lines 27–35). Finally, the algorithm returns the temporal path $\mathcal{P}$ in Line 36. On the absence of a temporal path, i.e., $v_d \notin \mathbb{V}$, Algorithm 4.2 returns $\varnothing$ (Line 38).

Consider an example temporal network shown in Figure 4.3 with the snapshots of graphs from six time instances $0 \leq t \leq 5$. Assume that at time $t = 0$, node a has a packet that needs to be delivered to node c. In an SD-DTN setting, node a sends a flow-rule request to the controller for a STORE/FORWARD decision. Upon receiving the request, controller makes use of the predicted graph sequence $\mathcal{S}'_0 = \langle G'_1, G'_2, G'_3, G'_4, G'_5 \rangle$ and estimates a possible temporal path, computes the network control decision, and sends the decision to node a. As per Algorithm 4.2, the triple (h_v, p_v, c_v) associated with each node is initialized and the subsequent changes in the values of triples are provided in

Algorithm 4.2: Earliest Arrival Path with Minimal Storage Time (EAPMST).

Data:
$\mathcal{S}'_t$ — Graph sequence $\mathcal{S}'_t = \langle G'_{t+1}, G'_{t+2}, \ldots, G'_{t+M} \rangle$
$\mathbb{V}$ — Set of visited nodes
h_v — Minimum distance (or number of hops) from source node to node v
p_v — Predecessor node of node v
c_v — Cumulative buffering time at nodes in reaching node v
$G_k = (V_k, E_k)$ — Graph at k^{th} time instant and ($V_k = V$ for all k)
$\mathcal{N}_v^{G_k}$ — Set of neighbors of node v in graph G_k

foreach $v \in V \setminus v_0$ **do** ▷ Initialize triple of each node
  $h_v \leftarrow \infty;\ p_v \leftarrow \varnothing; c_v \leftarrow \infty$
end
$h_{v_0} \leftarrow 0;\ p_{v_0} \leftarrow v_0;\ c_{v_0} \leftarrow 0;\ \mathbb{V} \leftarrow \{v_0\};\ k \leftarrow 1$
▷ Exploration phase
while $v_d \notin \mathbb{V} \wedge k \leq M$ **do**
  $\mathbb{V}' \leftarrow \varnothing$
  foreach $v \in \mathbb{V}$ **do** ▷ Visit the unvisited neighbors of visited nodes
    foreach $u \in \mathcal{N}_v^{G_k} \setminus \mathbb{V}$ **do**
      if $c_v < c_u$ **then**
        $h_u \leftarrow h_v + 1;\ p_u \leftarrow v;\ c_u \leftarrow c_v$
      end
      $\mathbb{V}' \leftarrow \mathbb{V}' \cup \{u\}$
    end
  end
  foreach $v \in \mathbb{V}$ **do**
    $c_v \leftarrow c_v + 1$ ▷ Update c_v of visited nodes in $\mathbb{V}$
  end
  $\mathbb{V} \leftarrow \mathbb{V} \cup \mathbb{V}'$ ▷ Update the set of visited nodes
  $k \leftarrow k + 1$
end
▷ Generate the temporal path by retracing from v_d
if $v_d \in \mathbb{V}$ **then**
  $\mathcal{P}' \leftarrow \langle\!\langle\, v_d \,\rangle\!\rangle;\ v \leftarrow v_d$
  ▷ Retrace path from v_d to v_0 using the triple associated with the nodes
  while $v \neq v_0$ **do**
    $\mathcal{P}' \leftarrow p_v \triangleright \mathcal{P}'$ ▷ Prepend p_v to path $\mathcal{P}'$
    $v \leftarrow p_v$
  end
  $\mathcal{P} \leftarrow \langle\!\langle\, \rangle\!\rangle;\ k \leftarrow 1;\ v_i \leftarrow v_0$
  ▷ Generate temporal path $\mathcal{P}$ from v_0 to v_d with w_is
  foreach $v_j \in \mathcal{P}'$ **do**
    $w_i \leftarrow 0$ ▷ w_i: Buffering time at node v_i
    while $(v_i, v_j) \notin E_k$ **do**
      $k \leftarrow k + 1;\ w_i \leftarrow w_i + 1$
    end
    $\mathcal{P} \leftarrow \mathcal{P} \triangleleft v_i(w_i)$ ▷ Append $v_i(w_i)$ to path $\mathcal{P}$
    $v_i \leftarrow v_j$
  end
  return $\mathcal{P}$
else
  return $\varnothing$ ▷ Temporal path does not exist
end



(a) $t = 0$ (b) $t = 1$ (c) $t = 2$

(d) $t = 3$ (e) $t = 4$ (f) $t = 5$

Figure 4.3: Example temporal graph sequence for six time instances. At $t = 0$, node a is assumed to have a packet destined to node c.

Table 4.1: Triple values (h_v, p_v, c_v) of each node at different time instances.

		Node					
		a	b	c	d	e	f
Time	$t = 0$	$(0, a, 0)$	$(\infty, \varnothing, \infty)$	$(\infty, \varnothing, \infty)$	$(\infty, \varnothing, \infty)$	$(\infty, \varnothing, \infty)$	$(\infty, \varnothing, \infty)$
	$t = 1$	$(0, a, 1)$	$(1, a, 0)$	$(\infty, \varnothing, \infty)$	$(\infty, \varnothing, \infty)$	$(\infty, \varnothing, \infty)$	$(1, a, 0)$
	$t = 2$	$(0, a, 2)$	$(1, a, 1)$	$(\infty, \varnothing, \infty)$	$(\infty, \varnothing, \infty)$	$(1, f, 0)$	$(1, a, 1)$
	$t = 3$	$(0, a, 3)$	$(1, a, 2)$	$(\infty, \varnothing, \infty)$	$(2, e, 0)$	$(1, f, 1)$	$(1, a, 2)$
	$t = 4$	$(0, a, 4)$	$(1, a, 3)$	$(2, b, 2)$	$(2, e, 1)$	$(1, f, 2)$	$(1, a, 3)$

Table 4.1. At $t = 1$, node a comes in contact with nodes b and f so that the packets can be sent to both the nodes in a single hop without incurring any buffering time at node a. Upon visiting nodes b and f, their triples are updated to $(1, a, 0)$ (Lines 7–12). Further, the buffering times of already visited nodes are incremented by 1 (Lines 15 and 16). At time $t = 2$, node b becomes isolated and node f comes in contact with e, i.e., node f can transfer the packet to e without any buffering. Node e can further transfer the packet to node d when the link gets established between them at time $t = 3$ while node b stores the packet due to its isolation. At $t = 4$, nodes a and b get connected to destination c,

thereby, marking node c as visited. However, there exist two temporal paths $\langle\!\langle\, a(3), c \,\rangle\!\rangle$ and $\langle\!\langle\, a(0), b(2), c \,\rangle\!\rangle$ with cumulative buffering times 3 and 2 time units, respectively. Our algorithm chooses the path $\langle\!\langle\, a(0), b(2), c \,\rangle\!\rangle$ due to its lowest cumulative buffering time. We stop the algorithm at $t = 4$ to ensure the earliest delivery even though there exists a temporal path $\langle\!\langle\, a(0), f(0), e(0), d(1), c \,\rangle\!\rangle$ with cumulative buffering time of only 1 time unit at $t = 5$.

Even though Algorithm 4.2 estimates the temporal path to the destination, *the path is used only for computing the* STORE/FORWARD *decision at the node (switch) which asks the controller for the flow-rule (in our example, node a) at time $t = 0$.* At time $t = 1$, the original graph G_1 becomes available and the network sequence $\mathcal{S}'_t$ can be re-computed using Eqn. (4.1) with the recent states of edges, i.e., the values of $Q_{(u,v),1}$, available from G_1. In other words, the temporal path computed at time $t = 0$ is valid only for $t = 0$ and can be re-computed at any time instant. In our SD-DTN framework, the graph sequence prediction (Algorithm 4.1) and EAPMST (Algorithm 4.2) are realized as SDN-controller applications (as depicted in Figure 4.5) to arrive at the STORE/FORWARD decision. For the temporal path $\mathcal{P} = \langle\!\langle\, v_0(w_0), v_1(w_1), \ldots, v_d \,\rangle\!\rangle$, the network control decision D or the flow-rule action is computed as

$$D = \begin{cases} \text{STORE (HardTimeout = } w_0) & \text{if } w_0 > 0, \\ \text{FORWARD} & \text{otherwise.} \end{cases} \tag{4.3}$$

If the temporal path requires the packets need to be buffered at the source node v_0 (i.e., $w_0 > 0$), the flow-rule is sent with its action as STORE. The time units to store the packets of that flow is realized using the HardTimeout attribute of flow-rules as described in Section 4.3. In the absence of a temporal path, i.e., $\mathcal{P} = \varnothing$, the packet is STORE-ed for a default time unit. We select the default time unit as the average time we observed for a network topology change. Since we limit the number of predictions to M, i.e., $|\mathcal{S}'_t| = M$, the worst-case time complexity of Algorithm 4.2 is estimated to be $\mathcal{O}(M|V|^2)$.

4.3 Realizing the STORE Action for SD-DTNs

In order to realize the controlled buffering operation in SD-DTNs, we propose a new action STORE which enables the SDN controller to dictate the switches to buffer packets for a specified unit of time. However, the existing SDN architecture is integrated with actions only for the controlled forwarding and dropping of packets. Therefore, in SD-DTN, the

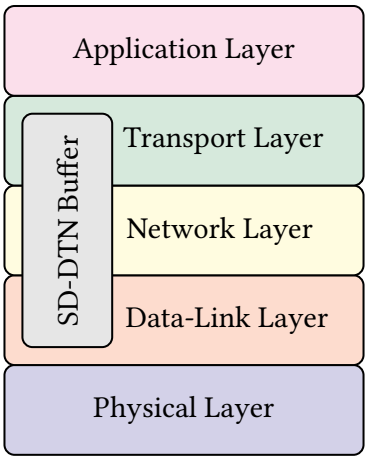


Figure 4.4: Protocol stack for the SD-DTN switch with a dedicated buffer for controlled buffering.

STORE action requires a mechanism for the controllers to specify the duration of buffering. On the other hand, SD-DTN switches require a new design capable of buffer management during the STORE action. At present, OpenFlow enables the controller to control the network flows using the attributes of Layers 2, 3, and 4 of the TCP/IP protocol stack. Therefore, for the controlled buffering, we define an SD-DTN buffer spanning the data-link, network, and transport layers of the TCP/IP protocol stack as shown in Figure 4.4 and the packets of different flows can be buffered using the attributes of Layers 2–4. That is, the STORE action is independent of the match-fields and can be used with the existing match-fields from Layers 2–4 of the OpenFlow protocol. In our experiments, we used the attribute of Layer 3, i.e., the destination IP address of the packets, for defining and controlling the network flows. Due to such a non-dependence on the attributes constituting the match-fields, the STORE action can capture any future improvement/addition of fields in the OpenFlow protocol. On the other hand, the buffering time of packets stored using the STORE action can be specified using the HardTimeout attribute of the flow-rule as discussed in Section 4.3.1.

4.3.1 Buffering Time in SD-DTN

As mentioned in Section 4.2, it is an important question in SD-DTN that for *how much time* the packets of a network flow need to be stored in an SDN-switch during link disruptions. The proposed EAPMST algorithm answers the question by providing a temporal path with required buffering time at each node. That is, at time $t = 0$, the path $\mathcal{P} = \langle\!\langle\, v_0(w_0), v_1(w_1), \ldots, v_d \,\rangle\!\rangle$ suggests that the packets entering node v_0 need to be

buffered for w_0 time units after which a link toward v_1 is expected. It is important here to note that not each packet entering node v_0 is buffered for w_0 time units. Instead, *the buffering time is specified for the network flow.* In SD-DTN, a flow-rule involving the STORE action with an associated buffering time of w_0 time units is interpreted as follows: *for the next w_0 time units, the packets corresponding to the network flow (i.e., those matched with the flow-rule) need to be buffered within the switch's buffer.* For example, assume a flow-rule involving a STORE action with an associated buffering time of 10 seconds. For a network flow with packets' inter-arrival time of 1 second, the first packet is buffered for 10 seconds, the second packet for 9 seconds, the third packet for 8 seconds, and so on. In other words, the flow-rule is valid only for w_0 time units (in the above example, for 10 seconds) and needs to be replaced with a newly computed flow-rule from the controller. Such a life-span or validity time for a flow-rule can be specified by exploiting the HardTimeout attribute of the flow-rules. The HardTimeout specifies the life-time of a flow-rule irrespective of a packet's match with that rule. Therefore, the SDN controller applications can issue flow-rules with STORE actions by specifying the required buffering time as the HardTimeout attribute of the flow-rule. In other words, we map the buffering time required for a flow to the validity time of the corresponding flow-rule.

4.3.2 A New Switch Prototype for SD-DTN

In order to realize the proposed controlled buffering in SD-DTN, the switches need to be capable of managing the buffered packets according to the instructions from the controller. Apart from buffering packets, the switches should inform the controller on the expiry of flow-rules, especially the ones that involve a STORE action. Due to the lack of such capabilities in the existing SDN switches for meeting the requirements of SD-DTNs, we propose a new architecture for SD-DTN switch as provided in Figure 4.5.

The SD-DTN switch consists of a *Packet Handler* to receive packets from the network and push them to the flow-table pipeline. Further, the packets are matched with the existing flow-rules. Upon a rule-hit event with a flow-rule involving the FORWARD action, the packets are forwarded to the port specified in the action. On the contrary, a STORE action prompts the flow-table handler to push the packets to the *Storage System* where the packets are buffered in the SD-DTN buffer. The *Storage System* is equipped with a *Buffer Manager* to decide on the type of memory to store the packets. In case the buffering time w_0 is larger than a threshold ξ, the packets are stored in the secondary storage (denoted as HDD in Figure 4.5) while a STORE action with less buffering time keeps the

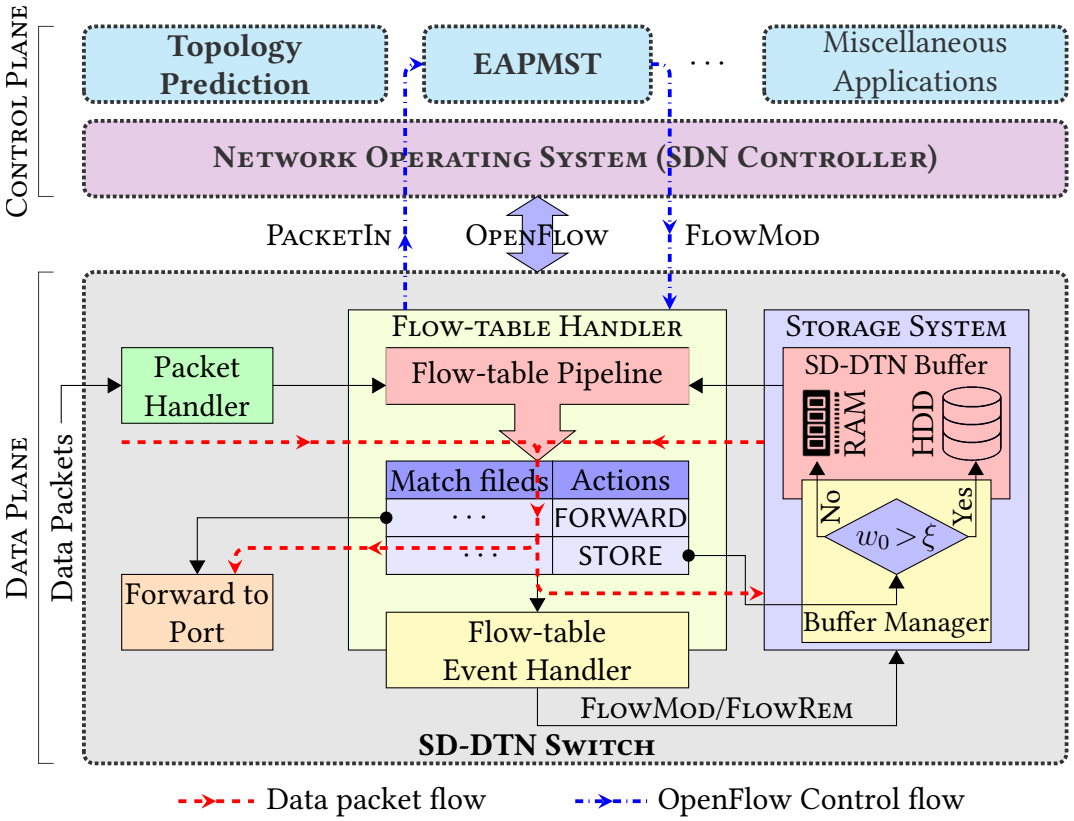


Figure 4.5: A new architecture for SD-DTN switch and its interaction with the SDN controller. The Red dashed lines show the flow of data packets within the switch while the Blue dash-dotted lines show the flow of SDN control packets between switch and the controller.

packets within the primary memory or ternary content addressable memory (denoted as RAM in Figure 4.5).

Upon the expiration of a flow-rule involving the STORE action, the buffered flow needs to be rechecked for a new rule in the flow-table or prompts the switch to request the controller for a modified flow-rule. The *flow-table handler* is equipped with a *flow-table event handler* to notify the *Storage System* on the removal or addition of flow-rules so that the stored packets can be forwarded to the flow-table pipeline. It is possible that an incoming packet may result in a modified flow-rule in the flow-table which can be used for forwarding the buffered packets of that flow. In case of a flow-rule miss, the SD-DTN switch generates a PacketIn message with the packet and sent to the controller. The flow of data packets within the SD-DTN switch is shown using the Red dashed lines in Figure 4.5. The controller receives and forwards the PacketIn to the *EAPMST* controller application, which computes the flow-rule based on the packet attributes and the network

topology from *Topology Prediction* application. Further, the flow-rule is embedded in a FlowMod message and sent to the switch. Upon getting the notification of a flow-rule addition, the *Storage System* pushes the buffered packets to the flow-table pipeline and action corresponding to the new flow-rule is taken on them. The Blue dash-dotted lines in Figure 4.5 depict the flow of SDN control packets between the switch and the controller.

Due to the dynamism in network topology, SD-WNs are required to frequently update the flow-rules at the switches. In case of SD-DTNs, the packets are be buffered during link disruptions in the flow-paths and forwarded when the links become alive. Our proposed EAPMST algorithm controls the buffering time of packets at each switch by making use of the predicted network topology. The efficacy of our SD-DTN framework with the controlled buffering mechanism and the performance of EAPMST algorithm are tested on an SD-WMN testbed involving link disruptions as described in Section 4.4.

4.4 Experimental Setup for SD-DTNs

For analyzing the proposed SD-DTN framework, we created a disruption-prone SD-WMN involving the following three mobility patterns for the nodes: (*i*) *stationary mobility*, (*ii*) *group mobility*, and (*iii*) *random mobility*. The testbed is deployed in the academic department building of Indian Institute of Space Science and Technology (IIST), Avionics Block, and spans multiple floors (Floor-0 and Floor-1). The layout of the testbed from Floor-0 of the building is shown in Figure 4.6.

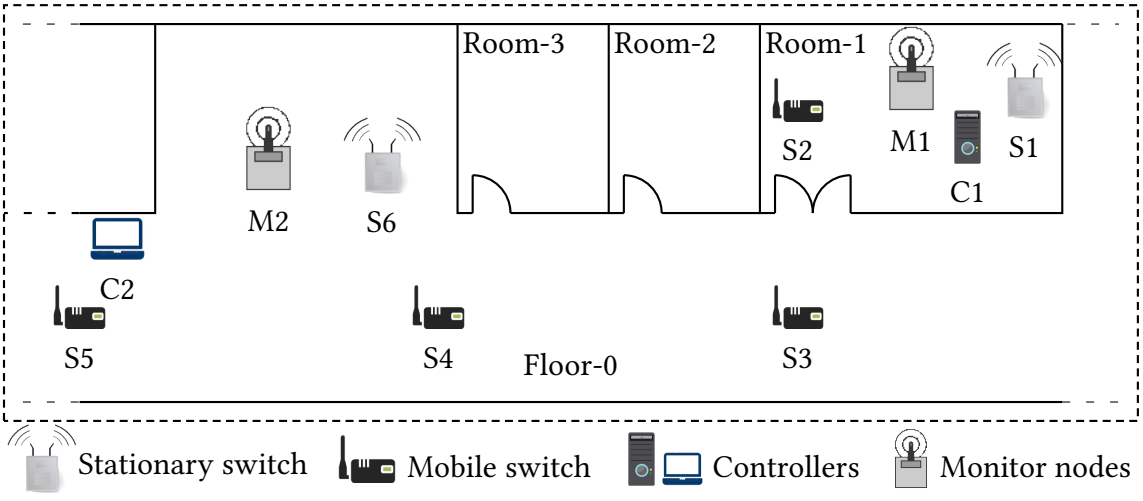


Figure 4.6: Layout of the SD-DTN testbed from Floor-0 of the IIST Avionics Department Building.

Table 4.2: Hardware and software specifications of the SD-DTN testbed.

		Specification
Controller	Hardware	C1: HP EliteDesk 800 G1 Tower PC Intel Core i5-4570 CPU @ 3.20 GHz, 4 GB RAM
		C2: Dell Vostro 2520 Laptop Intel Core i3-3110M CPU @ 2.40 GHz, 4 GB RAM
	Operating System	C1: Ubuntu 18.04 LTS, C2: Ubuntu 14.04 LTS
	Routing Protocol	Optimized Link State Routing (OLSR) v0.6.5.4
	SDN Controller	POX 0.3.0 (Dart)
	Southbound Interface	OpenFlow v1.0
Switch	Hardware	Raspberry Pi 3 Model B, Quad Core CPU@1.2GHz Broadcom BCM2837 (64bit), 1 GB RAM
	Operating System	Raspbian Stretch Lite
	Routing Protocol	Optimized Link State Routing (OLSR) v0.6.5.4
	Switch	Open vSwitch v1.5.0 (with OpenFlow v1.0)
Monitor	Hardware	Alix3d3 AMD Geode LX800 @ 500 MHz, 256 MB RAM Wireless Card: Wistron DNMA92 802.11 a/b/g/n miniPCI WiFi module
	Operating System	Voyage Linux-0.9.2

The network consists of six SD-DTN switches (S1–S6) in data plane with two physical controllers (C1 and C2) in control plane. Among the six switches, S1 and S6 are stationary and are deployed in Room-1 and the corridor, respectively. On the other hand, switches S2, S3, S4, and S5 are mobile nodes and follow respective mobility models through Floor-0 and Floor-1 during the experiments. Nodes C1 and C2 constitute the SDN controllers which are fixed in Room-1 and the corridor, respectively. We use multiple physical controller approach to address the single point-of-failure problem in SDN resulting from the possible controller disconnections during network partitioning. Therefore, the controllers are placed in such a way that all switches get connected to the control plane either through C1 or C2 in the experimental zone. Two monitor nodes, M1 and M2, are used to capture the network traffic for the later analysis of network performance. The SDN control channel is operated in in-band mode with SD-OLSR as the communication protocol. Switches use the network topology from SD-OLSR and perform switch-initiated handoff to the nearest controller during their mobility. The hardware and software specifications used for the SD-DTN testbed are shown in Table 4.2.

4.4.1 Mobility Models for the Experiments

In order to realize link disruptions in the experimental SD-DTN testbed, we make the SD-WMN switches move under the three mobility models during the experiments: (*i*) stationary, (*ii*) group, and (*iii*) random. The changes in network topology for the three mobility models during the timeline of our experiments are shown in Figure 4.7.

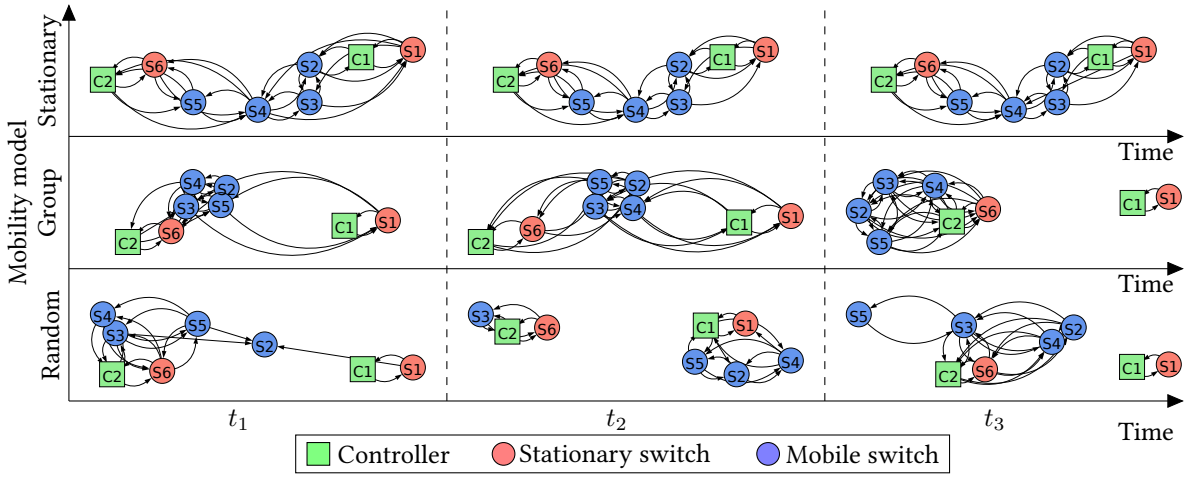


Figure 4.7: Snapshots of network topology from the three mobility models during the experiment timeline.

4.4.1.1 Stationary Mobility Model

In stationary mobility model, SDN switches remain stationary at their fixed locations as in the testbed layout shown in Figure 4.6. Therefore, the network topology remains almost the same and connected throughout the experiments. Even though the nodes are stationary, the network is prone to link disruptions, as revealed from Figure 4.8(a), due to environmental conditions such as the presence of machines and walls, and the movement of people. The switches communicate to the controller as well as between each other through multihop paths. Stationary mobility model is a popular form of network deployment where the switches remain fixed to provide a wireless mesh backhaul [22].

4.4.1.2 Group Mobility Model

In case of group mobility, the mobile switches, i.e., S2, S3, S4, and S5, form a group and move together in the experimental zone spanning Floor-0 and Floor-1 of the department building. Since they move as a group, the mobile nodes form links between each other.

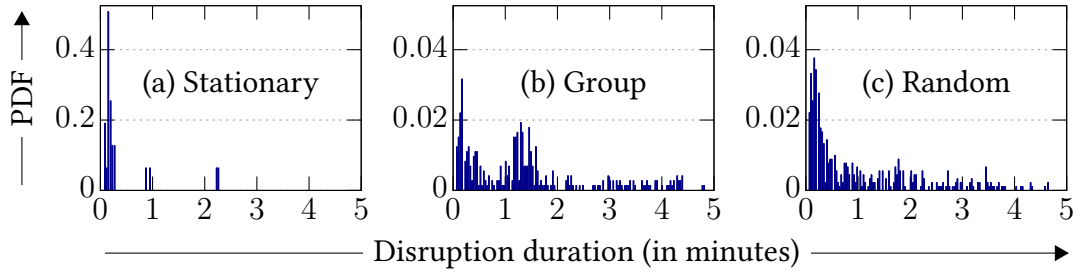


Figure 4.8: Distribution of link disruption durations, represented as Probability Density Functions (PDFs), from our experiments in SD-DTN testbed corresponding to the three mobility models.

However, the group gets disconnected from one controller as well as from the stationary switches as they move from one end of the building to the other end as shown in Figure 4.7. During the mobility period, the mobile switches get handed over to the nearest controller depending on the cost to reach both controllers. Group mobility can be observed among students in campus environments and in enterprises as well. The distribution of link disruption durations incurred in the group mobility model of our experiments is shown in Figure 4.8(b).

4.4.1.3 Random Mobility Model

In random mobility model, mobile switches are allowed to move at their discretion through the experimental zone. Due to the randomness in movement, the links between nodes get disrupted more often (see Figure 4.8(c)) and necessitate frequent controller handoffs. Also, the disruption-prone wireless links result in network partitions as shown in Figure 4.7. Examples of random mobility model include the mobility pattern of people in a city and the mobility of animals in wireless sensor networks deployed for animal monitoring.

The distribution of link disruption durations (represented as probability density functions in Figure 4.8) of stationary, group, and random mobility models from our experiments indicates that the network is prone to link disruptions even if the nodes remain stationary (see Figure 4.8(a)). Therefore, our proposed SD-DTN framework forms a suitable candidate for handling link disruptions even in stationary wireless network environments.

4.4.2 SDN Controller Application Schemes for the Experiments

Besides the three mobility models used for realizing link disruptions, we employ the following three SDN-based network control schemes: (*i*) SDN, (*ii*) SD-DTN, and (*iii*) SD-DTN+EAPMST, for testing the efficacy of the proposed controlled buffering mechanism. The procedures for the three SDN control schemes are described in Algorithm 4.3.

4.4.2.1 SDN Scheme

We use the traditional SDN-based network control as the benchmark scheme for comparing the performance of the proposed SD-DTN framework. In SDN scheme as described in Algorithm 4.3.1, the routing protocol for the network flows is administered from the controller where the flow-rules are derived from the shortest path between source and destination nodes. Upon receiving a PacketIn message from a switch, the controller extracts the source and destination of the flow and computes the Dijkstra's shortest path using the network topology at that instant (Line 1). If a path exists and the switch is an element in that path, the controller creates a flow-rule with FORWARD action (Lines 3 and 4). On the other hand, the rule is provided with a DROP action in case the switch does not take part in the shortest path (Line 6) or in the absence of a path. On a link failure, the scheme considers alternate paths for routing the packets while the packets are dropped at the switches during the absence of alternate paths. Such a dropping of packets makes the transport layer at the source node to reduce the sending rate. It is important to note that *no STORE action (i.e., no buffering) is involved in the SDN scheme.*

4.4.2.2 SD-DTN Scheme

In SD-DTN scheme, we introduce the proposed controlled buffering of packets at the switches. However, the default buffering time (or the HardTimeout of flow-rules) for the flows is set to 10 seconds. The buffering time is chosen due to the fact that the network changes its topology in every 10 seconds on an average in our experiments. In SD-DTN scheme as described in Algorithm 4.3.2, upon receiving a PacketIn message from a switch, the controller computes the Dijkstra's shortest path (Line 1) using the current network topology and the flow-rule is created with the FORWARD action if the switch participates in the shortest path (Lines 3 and 4). The packets are dropped in case the switch does not involve in the path (Line 6). In case there does not exist a path, the controller issues a flow-rule with a STORE action with HardTimeout attribute set to 10 seconds (Line 9).

Algorithm 4.3: Experimental SDN controller applications.

Data:
S_{in} — Switch that generates the PacketIn message
Pkt — Data packet contained within the PacketIn message
ShortestPath(a, b) — Function that returns Dijkstra's shortest path between nodes a and b based on ETX
FindTemporalPath(a, b) — Function that executes Algorithm 4.2 and returns the temporal path between nodes a and b
(HardTimeout is specified in seconds)

Algorithm 4.3.1: SDN

```
P ← ShortestPath(Pkt.src, Pkt.dst)
if P ≠ ∅ then
    if S_in ∈ P then
        Action ← FORWARD (PORT 1)
    else
        Action ← DROP
    end
else
    Action ← DROP
end
```

Algorithm 4.3.2: SD-DTN

```
P ← ShortestPath(Pkt.src, Pkt.dst)
if P ≠ ∅ then
    if S_in ∈ P then
        Action ← FORWARD (PORT 1)
    else
        Action ← DROP
    end
else
    Action ← STORE (HardTimeout=10)
end
```

Algorithm 4.3.3: SD-DTN + EAPMST

```
P =FindTemporalPath(S_in, Pkt.dst)
if P ≠ ∅ then
    if P.w_0 = 0 then
        Action ← FORWARD
    else
        Action ← STORE (HardTimeout=T.w_0)
    end
else
    Action ← STORE (HardTimeout=10)
end
```

Upon the expiry of flow-rules after 10 seconds, the stored packets are rechecked for new flow-rules as explained in Section 4.3.2.

4.4.2.3 SD-DTN+EAPMST Scheme

Here, we introduce the proposed EAPMST algorithm (Algorithm 4.2) to decide on the buffering time rather than assigning the default value of 10 seconds. The procedure for computing EAPMST-based buffering/routing decision is described in Algorithm 4.3.3. That is, upon receiving a PacketIn message, the *EAPMST* controller application estimates the temporal path $\mathcal{P}$ from current node v_0 to the destination using the predicted topology sequence from the *Topology Prediction* application (Line 1). If an appropriate link is available to forward the packet (i.e., $w_0 = 0$), a rule is issued with FORWARD as the action. On the contrary, if the path suggests a waiting time at v_0, i.e., $w_0 > 0$, a flow-rule is created with the STORE action and HardTimeout attribute set to w_0 seconds (Line 6). In case a temporal path does not exist, the packets corresponding to the flow are stored for the default time of 10 seconds (Line 9).

4.4.3 Application Protocols and Experiment Pattern

In order to generate real-world communication traffic, we used the following four protocols: (*i*) Hypertext Transfer Protocol (HTTP), (*ii*) File Transfer Protocol (FTP), (*iii*) Voice over Internet Protocol (VoIP), and (*iv*) Internet Control Message Protocol (ICMP). Among the four protocols, HTTP and FTP use reliable transport protocols in the transport layer while VoIP and ICMP use unreliable protocols. The experiments are carried out using all combinations of mobility models, SDN control schemes, and communication protocols. The detailed strategy of the experiments is shown in Figure 4.9.

Each experiment is designed for a duration of 10 minutes. When the network is up, each switch generates communication sessions with random destinations using the four protocols one after the other. Within an experiment, the sessions for HTTP, FTP, and ICMP are limited to 2 minutes. The data-rate of ICMP session is chosen between 1 and 10 packets/second at random. VoIP session is generated as per the G.711 encoding specifications with a data-rate of $64\,\mathrm{kbps}$. Such a high data-rate VoIP traffic creates significant overhead on SD-WMNs operating under in-band control. Subsequently, more number of SDN control packets from different VoIP sessions may overload the SDN control channel and lead to a network deadlock, thereby, degrading the network performance as observed in [21, 22, 27, 28]. In order to avoid such a deadlock in our experiments, the duration of

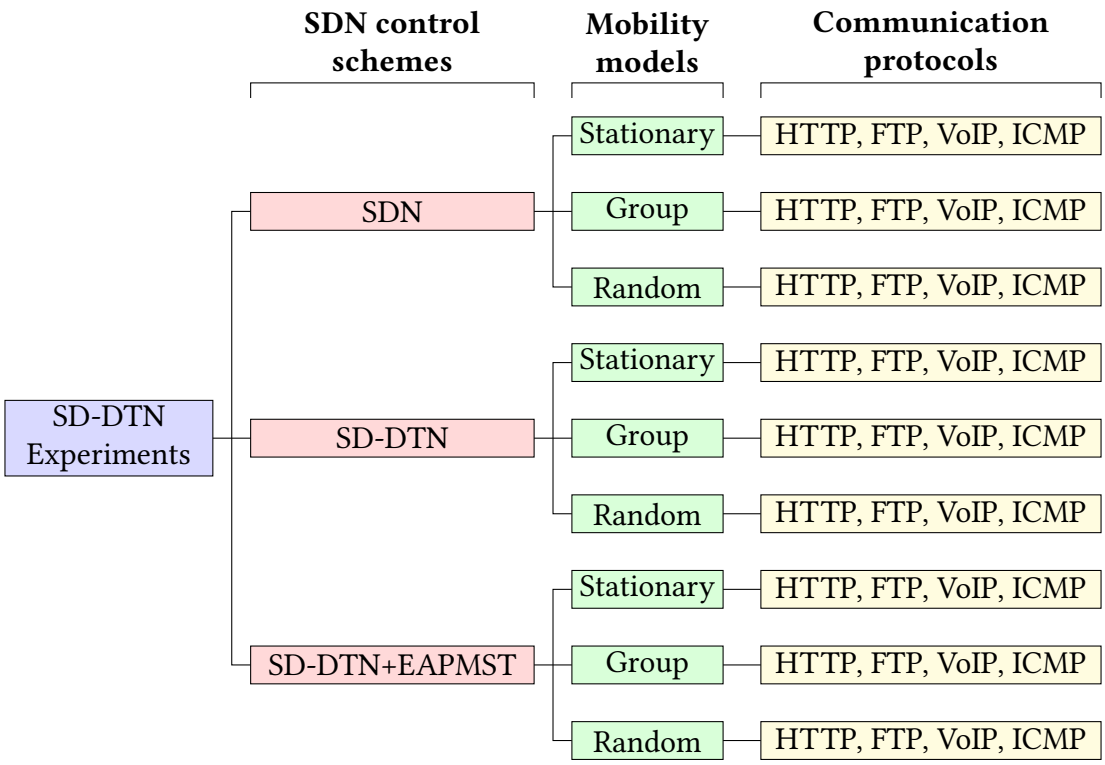


Figure 4.9: Experiment strategy for SD-DTN performance studies.

VoIP traffic is restricted to 1 minute.

Due to the instability of wireless links, we repeated each experiment 10 times and the efficacy of the proposed SD-DTN framework is analyzed using the average values of performance metrics over those 10 samples. Besides gathering the network traffic at all switches and controllers, we used two monitor nodes (M1 and M2) to capture the network traffic, which we use to synchronize the data collected from switches and controllers while analyzing the performance.

4.5 Performance Analysis

The primary objective of the proposed SD-DTN framework is to introduce a controlled buffering mechanism for the switches so that the packets of different flows can be stored at the switches during link disruptions in flow-paths and forward them to the destination when the links become alive. Such a controlled buffering can significantly improve the network performance especially in terms of end-to-end delay and throughput as discussed in Section 1.3. In order to measure the performance of SD-DTN, we define the following

six metrics: (*i*) *throughput*, (*ii*) *end-to-end delay*, (*iii*) *OpenFlow overhead*, (*iv*) *buffering time*, (*v*) *buffer occupancy*, and (*vi*) *prediction accuracy*.

4.5.1 Throughput

Throughput is considered as the most important measure in analyzing the performance of any network. Throughput measures the amount of data successfully transferred over the network under consideration. In SD-DTNs, the decision to forward or store the packets is administered from the controller and switches need to request the controller for the network control decision before acting on the packets. In addition, in-band control makes the control channel more constrained and forces the control packets traverse through multihop paths to reach the controller and switches, thereby, increasing the end-to-end delay and diminishing the throughput. Therefore, throughput forms an appropriate metric for SD-DTNs to capture the effectiveness of end-to-end communication sessions under dynamic network conditions.

Due to the nature of underlying transport protocols used in the four communication sessions, we define throughput separately for: (*i*) sessions that use reliable transport protocol and (*ii*) sessions that operate on unreliable channel.

Sessions that use reliable transport protocols: For protocols such as HTTP and FTP, which use reliable protocols in the transport layer, throughput is defined as the number of bytes successfully transferred over the network in unit time.

Sessions that use unreliable channel: For protocols such as VoIP and ICMP, which operate over unreliable protocols, throughput is defined in terms of *delivery ratio*, i.e., the ratio of the number packets delivered at the destination to the number of packets sent.

Figure 4.10 shows the throughput obtained from our experiments with all mobility models and SDN control schemes. As far as the reliable protocols are concerned, FTP offers better throughput compared to HTTP. Low throughput in HTTP is due to the significant number of TCP session control packets of small sizes generated from each web-page request. Consequently, the average frame sizes of FTP and HTTP sessions are observed to be 223 and 130 bytes, respectively. In terms of the SDN control schemes, SD-DTN+EAPMST dominates in performance in both HTTP and FTP. The default buffering time of 10 seconds leads to long TCP session time in SD-DTN, thereby, degrading the throughput. However,

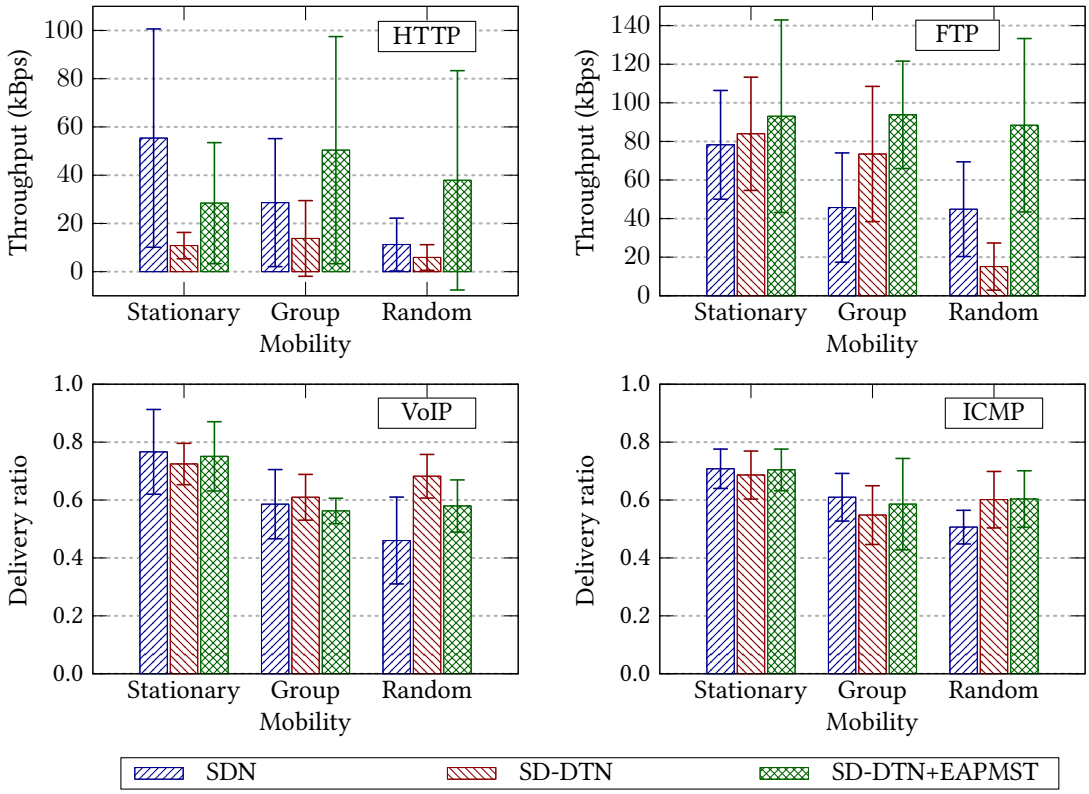


Figure 4.10: Throughput achieved by the four protocols under different mobility models and SDN control schemes.

the introduction of EAPMST significantly improves the throughput by tuning the buffering time depending on the network dynamics. Among the SDN and SD-DTN schemes, it is important to note that SD-DTN with default buffering time is better for FTP traffic while traditional SDN alone suffice for HTTP-based traffic.

In sessions such as VoIP and ICMP that use unreliable protocols, we can observe that the throughput reduces with increasing randomness in the mobility pattern. In stationary mobility model, traditional SDN performs as good as the proposed SD-DTN framework. However, in case of nodes involving mobility, SD-DTN dominates in performance, especially in random mobility model. The delivery ratio below 0.7 in group and random mobility models indicates further scope of improvement for SD-DTN as part of future work. Among the SDN control schemes, SD-DTN is more effective in VoIP (i.e., high data-rate traffic) than SD-DTN+EAPMST while ICMP performs better by making use of EAPMST rather than using SD-DTN alone. In short, the experimental results show that the proposed SD-DTN framework is a promising candidate for improving the throughput specifically in SD-WNs involving node mobility.

4.5.2 End-to-End Delay

One of the primary objectives of any next generation communication infrastructure is to reduce the end-to-end latency to the order less than 1 millisecond. Achieving latency to such a margin in DTNs while maximizing the delivery ratio poses a major challenge, especially in environments involving physical mobility of nodes and disruption of links. In this context, controlled buffering of packets at intermediate nodes provides an advantage in minimizing the end-to-end latency rather than dropping them during link disruptions or in the absence of source-destination paths as discussed in Section 1.3.

We define end-to-end delay as the difference in time at which the packet is delivered at the destination and the time at which it is sent from the source. Unlike traditional networks, SDN incurs an additional delay of switch-controller communication involved in the flow-rule request (PacketIn) and response (FlowMod) packets. Control channel delay, i.e., the delay incurred by the switches in receiving a FlowMod message on sending a PacketIn request, is more influential in in-band control where the same communication infrastructure is shared by the data and control packets. The delay incurred by the SDN control channel alone in our experiments is shown in Figure 4.11.

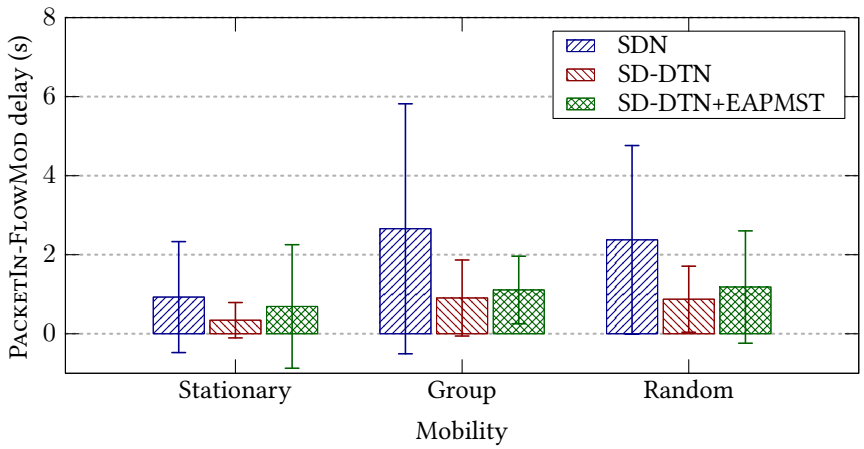


Figure 4.11: Average delay incurred in SDN control channel for the flow-rule request (PacketIn) and response (FlowMod) packets.

The end-to-end delays incurred by the data packets in the four communication sessions with the three SDN control schemes under different mobility models in our experiments are shown in Figure 4.12. In most cases, SDN scheme offers the least delay due to the fact that the packets are sent only in the presence of a source-destination path. That is, when the end-to-end path is disrupted, the packets are dropped. Among the communi-

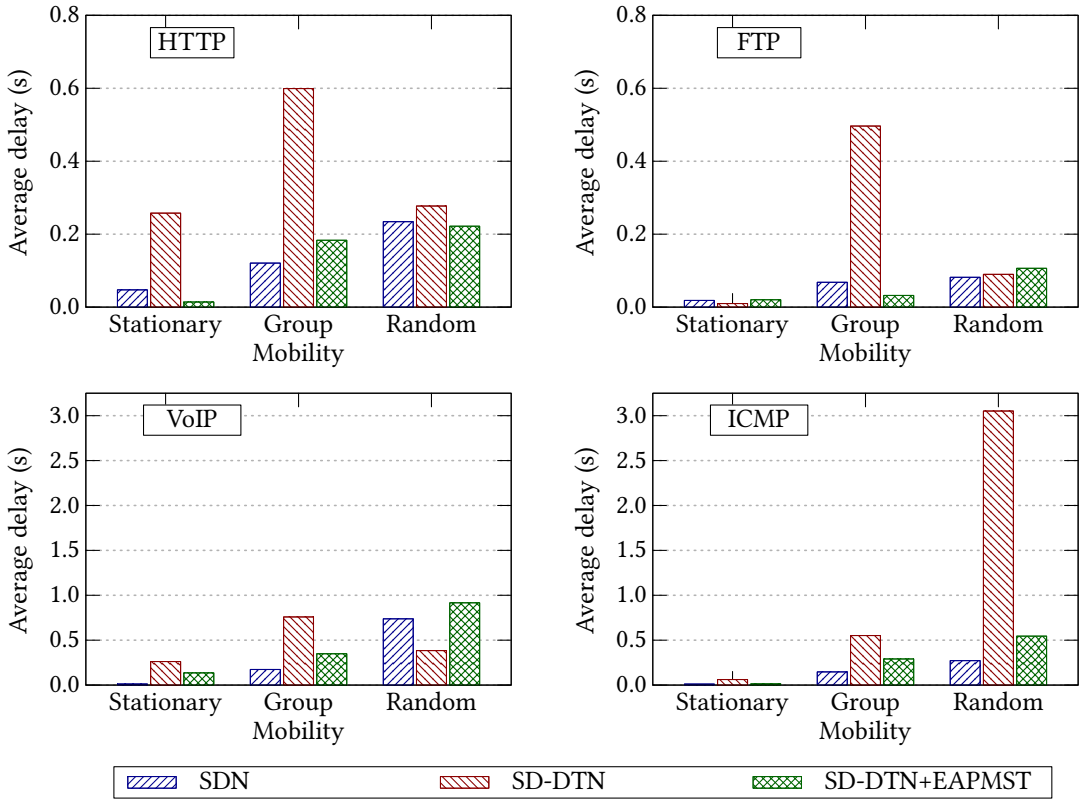


Figure 4.12: Average end-to-end delay incurred by the packets of the four communication protocols with the three SDN schemes under different mobility patterns.

cation sessions that use reliable protocols, FTP incurs the least delay. On the other hand, more number of web-page requests and associated small-sized TCP session control packets in HTTP incur delays in a wide range, thereby, increasing the end-to-end delay. The sessions with unreliable protocols also incur similar values of delay. In fact, we observe the highest average delay of 3.05 seconds in ICMP session during random mobility with the SD-DTN network control scheme. Moreover, the inherent rate control of TCP has a role in reducing the end-to-end delay of packets in HTTP and FTP compared to VoIP and ICMP.

Among the proposed schemes, SD-DTN with the default buffering time of 10 seconds incurs higher delay than SD-DTN+EAPMST. That is, the introduction of EAPMST with fine-tuned buffering time significantly reduces the delay as can be seen from Figure 4.12. It is important to note that EAPMST is capable of reducing end-to-end delay even below SDN scheme as visible in HTTP. As far as the mobility models are concerned, network with stationary nodes offers the least delay due to the stable network topology and resulting near-zero buffering times. Among the other two mobility patterns, the network under

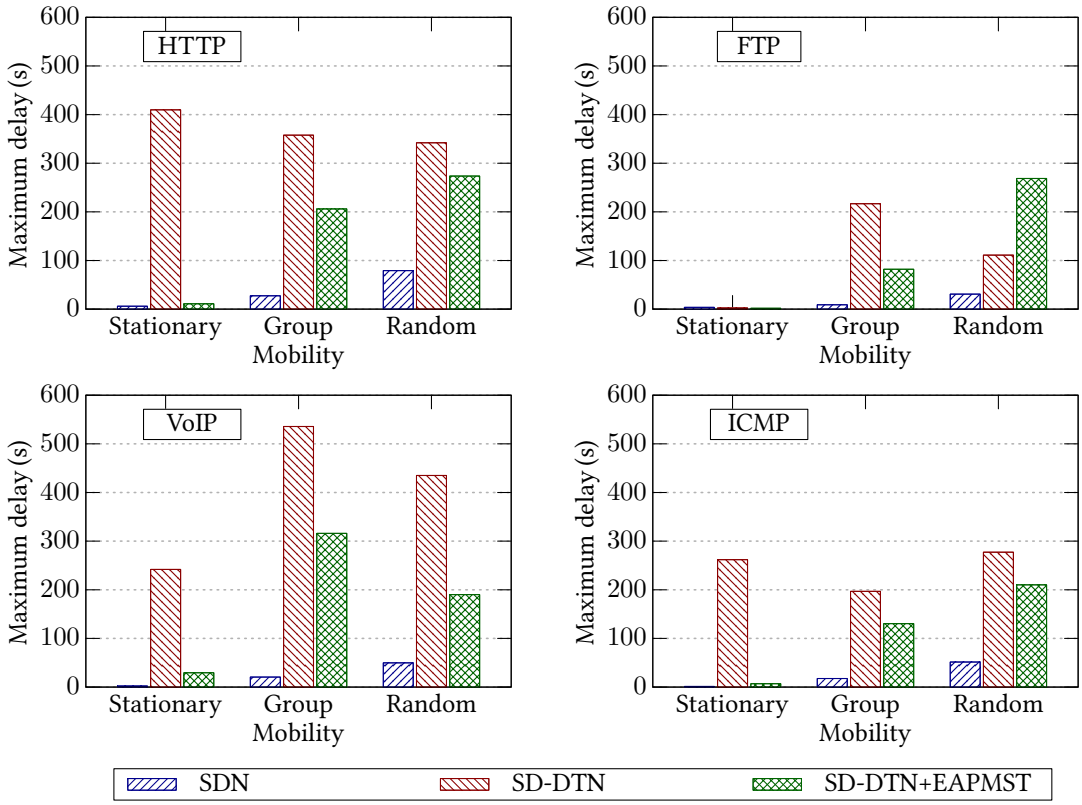


Figure 4.13: Maximum end-to-end delay incurred by packets of the four protocol sessions with the three SDN schemes under different mobility pattern of nodes.

group mobility offers better delay performance than nodes following random mobility due the stable links formed between the nodes forming the group.

Figure 4.13 shows the maximum delay incurred by the packets of the four communication sessions under different experimental settings. It is observed that the VoIP session suffers maximum delay of 535.67 seconds (8.9 minutes) under group mobility. Such a high delay demonstrates the efficacy of our SD-DTN framework in keeping the packets within the network during link disruptions for a long time and deliver them to the destination when the path for the flow gets re-established.

4.5.3 OpenFlow Overhead

Overhead defines the additional information that needs to be communicated over the network to successfully transmit the user application data. In SDN, the overhead is primarily due to the SDN control packets communicated between the switches and controllers. In software defined wired networks, switches request flow-rules from the controller upon

a flow-rule miss for a flow. Due to the static topology of wired networks, the flow-rule remains within the flow-table for a long time unless there exists an urgency to modify the rule. However, in case of dynamic wireless environments, the flow-rules need to be updated corresponding to the frequent topology changes especially if the flow-rule is a function of the network topology. Apart from such frequent SDN control packets, the *Storage System* in SD-DTN also prompts additional PacketIn messages on the expiry of a flow-rule to update the flow-table with the recent flow-rule. Therefore, it is important to measure the overhead required to operate the SD-DTN framework compared to the traditional SDN scheme. In addition, minimizing overhead is a challenging problem in SD-WNs that use in-band control channel.

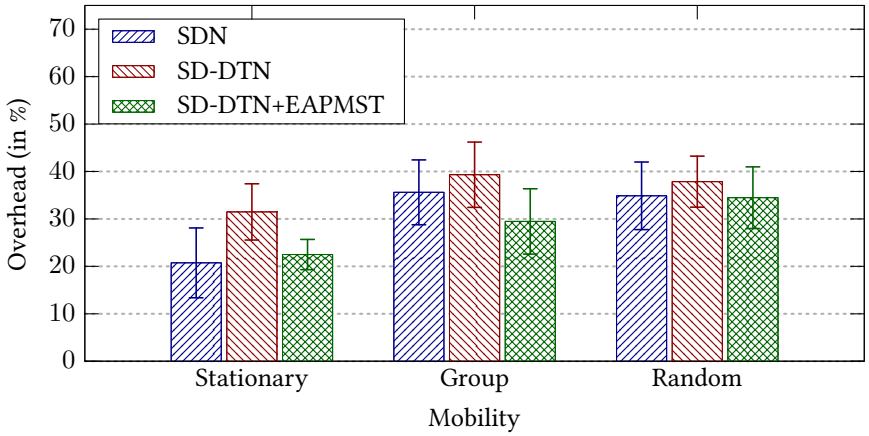


Figure 4.14: OpenFlow overhead on the network in the three SDN control schemes under different mobility models.

We define OpenFlow overhead as the ratio of OpenFlow traffic (in bytes) to the total traffic in the network (in bytes), measured in percentage terms. Figure 4.14 shows the OpenFlow overhead incurred by the network in different SDN control schemes under the three mobility models. As expected, SD-DTN offers the highest overhead in all mobility patterns due to the additional overhead from the *Storage System*. Moreover, the default buffering time (HardTimeout of flow-rules) of 10 seconds is also influential in increasing the overhead beyond 30%. Due to the in-band control channel, SDN control packets use multihop paths to reach the switches and controllers. At times, it may take a significant time for the flow-rules to reach the switches which may lead to indefinite storage time for the packets. Therefore, PacketIn messages are generated by the *Storage System* periodically apart from the ones resulting from the flow-rule deletion event. Unlike SDN and SD-DTN, introduction of EAPMST offers a clear advantage in bringing down the over-

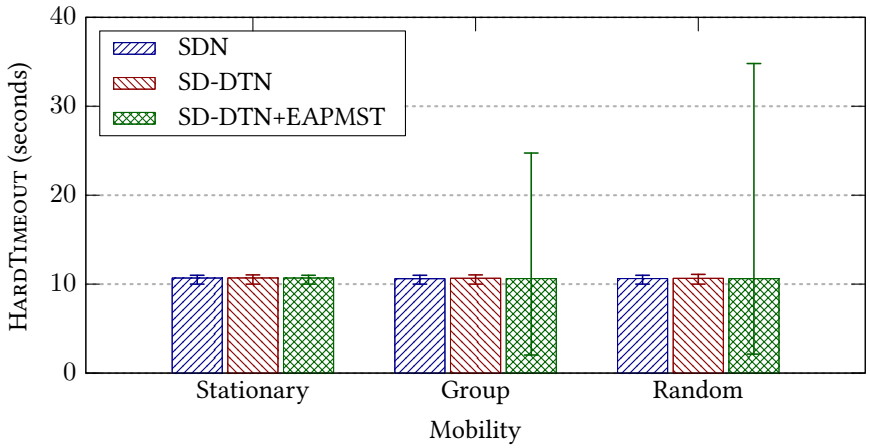


Figure 4.15: HardTimeout values provided by the three SDN control schemes under different mobility models considered. The error lines indicate the minimum and maximum values of the HardTimeout value, not the standard deviation.

head, even below the traditional SDN scheme, in group and random mobility patterns.

The range of HardTimeout (i.e., buffering time) values provided by the three SDN schemes is shown in Figure 4.15. The EAPMST scheme goes for a wide range of HardTimeout values derived from the predicted graph sequence while SDN and SD-DTN use the default value of 10 seconds. The maximum and minimum values of buffering time in EAPMST are observed to be 24.74 and 2.02 seconds, respectively, under group mobility. On the other hand, random mobility provides maximum and minimum HardTimeout of 34.81 and 2.13 seconds, respectively. That is, in case of sudden topology changes, EAPMST reduces the buffering time to a few seconds (less than the default value of 10 seconds) while slow topology changes are accommodated with long buffering times. Such a fine tuning of buffering time helps to reduce the overhead as visible in Figure 4.14.

4.5.4 Buffering Time

The time for which a packet is buffered within an SD-DTN switch has significant influence over the end-to-end delay, throughput, and the buffer space availability within a switch. More buffering time increases the end-to-end delay and makes less buffer space available within the switches. Therefore, we design EAPMST with the objective of minimizing the cumulative buffering time of packets at the intermediate nodes. For analyzing SD-DTN, we define buffering time as the average time a packet resides within a switch's buffer as the result of the STORE action. Figure 4.16 shows the average buffering time incurred by packets in two SDN control schemes under the three mobility models. Due to the absence

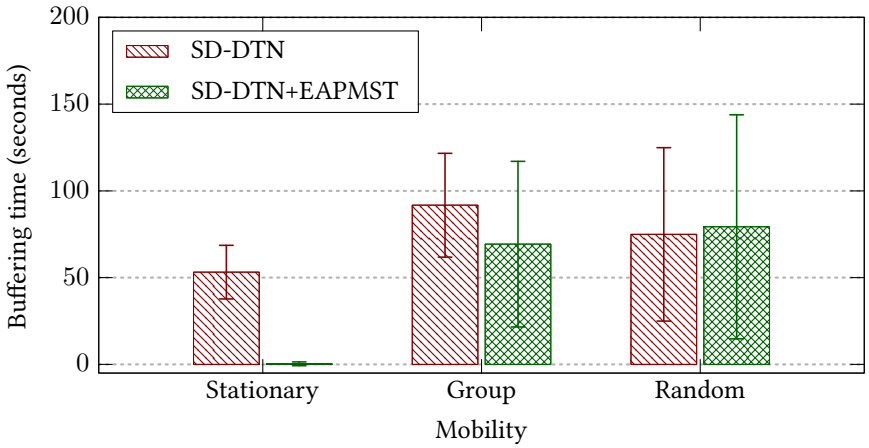


Figure 4.16: Buffering time incurred by packets in SD-DTN-based control schemes.

of controlled buffering, we discard SDN scheme from our analysis.

The stationary nodes offer the least buffering time due to the stable connected network topology, i.e., it is highly probable that there exists a path between any pair of nodes. However, at times buffering is required when links get disrupted among stationary nodes as shown in Figure 4.8. In such cases, SD-DTN buffers the packets for more time due to its default buffering time of 10 seconds. On the other hand, EAPMST predicts the topology with high accuracy and route the packets with minimum buffering time. Among the other two mobility patterns, SD-DTN+EAPMST performs better than using SD-DTN alone in group mobility while the reverse is true for nodes following random mobility patterns. Due to the presence of links between all mobile nodes, the behavior of network can be predicted more accurately in group mobility than the random mobility case. In other words, Figure 4.16 indicates the influence of the accuracy of prediction algorithm (Algorithm 4.1) in determining the buffering time as discussed in detail in Section 4.5.6.

4.5.5 Buffer Occupancy

Buffer occupancy is defined as the amount of buffer space (in bytes) used by the stored packets within the SD-DTN switch as the result of the STORE action. Buffer occupancy is an important metric of concern in cases where the flow-rule is a function of buffer space available within the switches. In SD-DTN, a STORE action in a flow-rule makes the packets of that flow get stored in the *Storage System*. Upon the removal of that flow-rule, the *Storage System* prompts the switch to request the controller for an updated rule. In case the updated rule has a FORWARD action (i.e., a link is re-established to forward the

packet), the stored packets of that flow are retrieved from the SD-DTN Buffer and forwarded through the specified link in order to free the buffer space. On the other hand, a STORE action in the rule forces the stored packets remain within the switch and, subsequent packets of that flow are also buffered in the SD-DTN buffer, thereby, increasing the buffer occupancy. It is important in any DTN to have adequate buffer space available so that sufficient packets can be stored in case of link disruptions and achieve maximum delivery ratio. Therefore, our EAPMST algorithm aims at minimizing the buffering time, thereby, freeing the buffer space for the packets of subsequent flows.

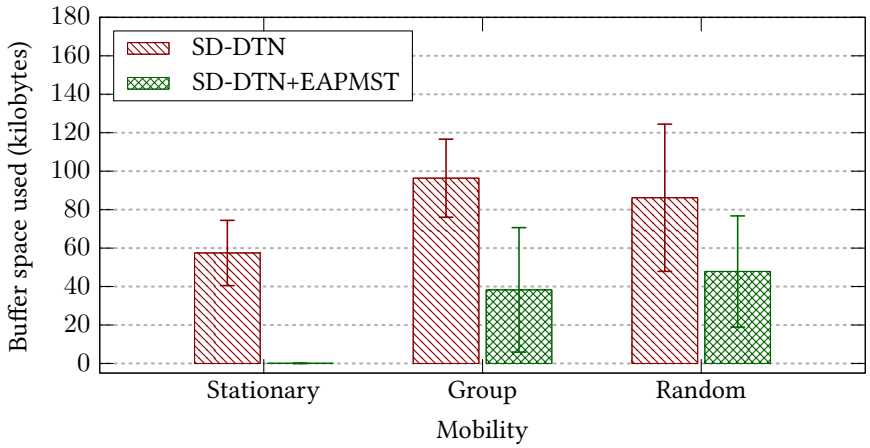


Figure 4.17: Buffer space occupied by SDN control schemes under the three mobility models.

Figure 4.17 shows the buffer space used by the packets in both SD-DTN and SD-DTN+EAPMST schemes under the three mobility models. The benchmark SDN scheme is discarded from analysis due to the absence of buffering. It is clear from Figure 4.17 that the introduction of EAPMST has a significant influence on reducing the buffer occupancy compared to the SD-DTN scheme, which buffers the packets in the absence of a path for the default time of 10 seconds. As far as the mobility models are concerned, stationary model uses the least amount of buffer space (less than 100 bytes) due to their connected topology. With the SD-DTN scheme, flows occupy 96.33 kB and 86.16 kB in group and random mobility, respectively. However, a reduction in buffer space usage in the order of 60.28% and 44.49%, respectively, for group and random mobility pattern is observed while using SD-DTN+EAPMST control scheme.

It is important to note that we cannot state a relationship between buffering time and buffer occupancy. That is, a small buffering time for a flow can result in significant buffer occupancy if the flow is of high data-rate such as VoIP. On the contrary, a low data-rate

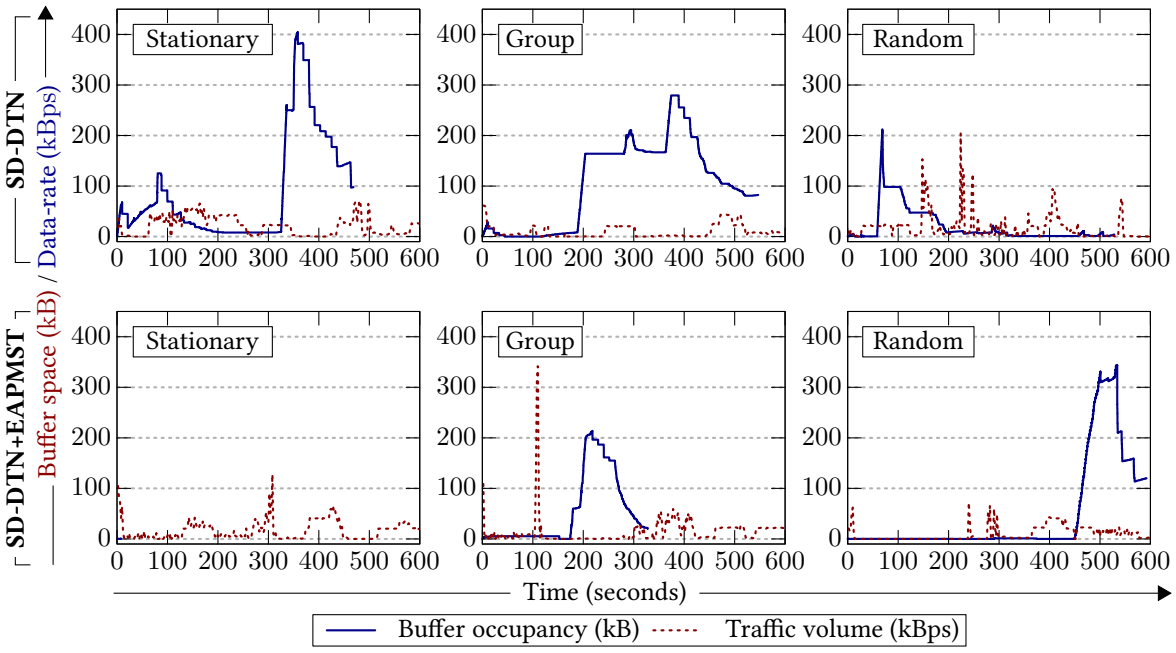


Figure 4.18: Buffer occupancy corresponding to the traffic volume through the timeline of the experiment on the most central node of the network.

flow (such as ICMP) may result in low buffer occupancy even though the matched flow-rule instructs for a long buffering time. Similar to buffering time, a relationship pattern is absent between buffer occupancy and traffic volume handled by a switch as shown in Figure 4.18. We analyzed the traffic volume on the most central node (computed in terms of the Betweenness Centrality [137] of the network) and corresponding buffer space used by the packets in SD-DTN and SD-DTN+EAPMST schemes.

Buffer occupancy increases only if a flow is subject to the STORE action and the increase can be more steep if the flow is of high data-rate. Also, at a particular time instant, different flow-rules may possess different actions. Among the rules, the ones with FORWARD action do not contribute to buffer occupancy even if the matched flow's data-rate is high. However, packets matched with a rule possessing the STORE action is forwarded to the *Storage System* irrespective of the data-rate.

4.5.6 Prediction Accuracy

Accuracy of graph prediction (Algorithm 4.1) is one of the most important factors that influences the performance of the proposed SD-DTN framework since EAPMST, the network control algorithm which estimates the buffering time, uses the predicted graph se-

quence $\mathcal{S}'_t$ from Algorithm 4.1 as its input. Therefore, we compute prediction accuracy to analyze the effectiveness of network control (STORE/FORWARD) decision in SD-DTNs. As discussed in Section 4.1, we use the t-step transition probability of Markov chains to predict the future graph sequences. For the analysis, we define prediction accuracy as the ratio of the number of edge states (either the edge's presence or absence) in G_{t+i} correctly predicted in G'_{t+i} to the total number of edges possible in the network.

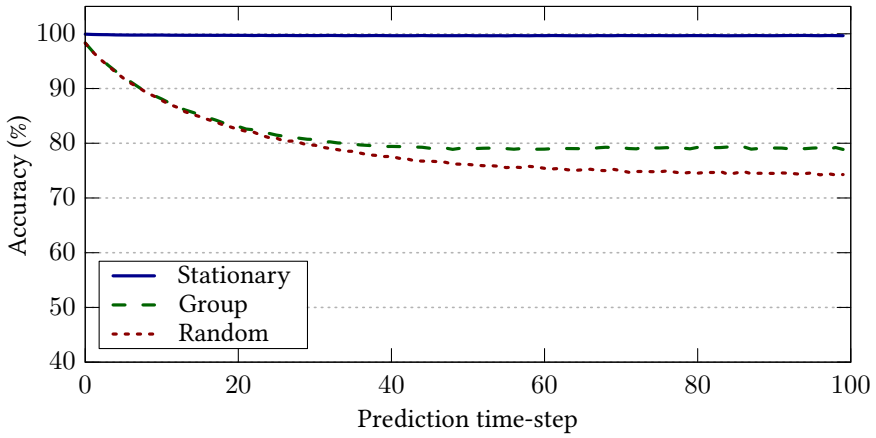


Figure 4.19: Prediction accuracy obtained in the three mobility models.

Figure 4.19 shows the accuracy obtained from our topology samples while predicting the graph instances for 100 future time instances. As expected, Algorithm 4.1 provides the highest accuracy beyond 99.6% for the stationary network due to the connected and stable network topology. Since the mobile nodes that form the group always possess links between each other, group mobility achieves accuracy better than the case of nodes following the random mobility pattern. On the whole, our prediction approach provides accuracy beyond 70% for all mobility models even for 100 future time-steps. However, the accuracy shows a decreasing trend with increasing time-steps.

4.5.6.1 Impacts of Prediction Accuracy

In order to assess the impact of decreasing prediction accuracy along the future timeline, we analyze end-to-end delay and buffering time through estimation using our experiment data. For estimation, we created a communication scenario where one message, whose source and destination are chosen at random, is generated at each time instant and routed using the proposed EAPMST algorithm (Algorithm 4.2). Figure 4.20 shows the end-to-end delay and corresponding buffering time achieved by the SD-DTN framework with

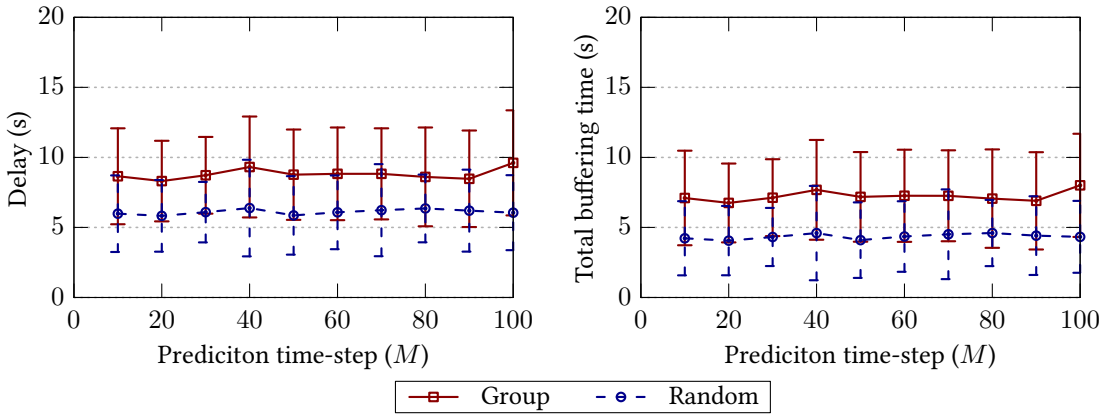


Figure 4.20: End-to-end delay and the corresponding buffering time achieved by SD-DTN with the default buffering time of 10 seconds.

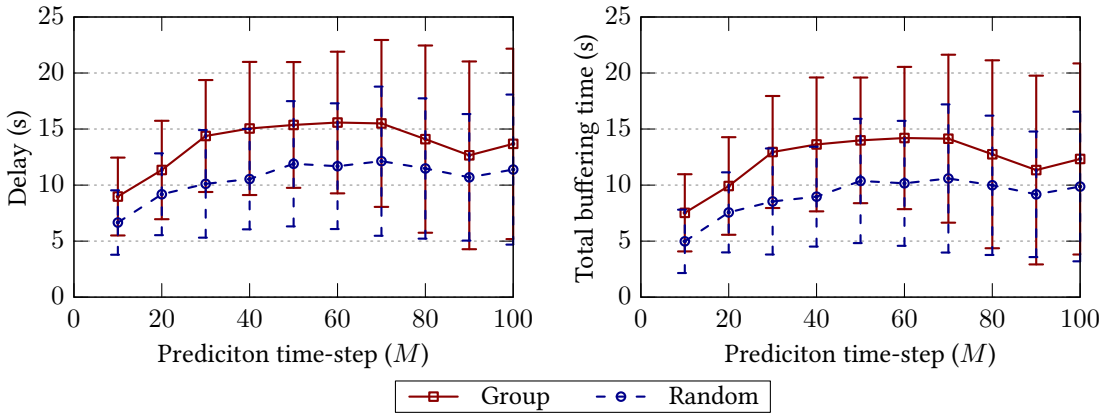


Figure 4.21: End-to-end delay and the corresponding buffering time achieved by SD-DTN with the default buffering time of M seconds.

the default buffering time of 10 seconds *in the absence of a temporal path*. It is important to note that the packets incur almost constant delay in both group and random mobility when the sequence is predicted for different future time-steps (M). Moreover, the major portion of the delay is attributed to the buffering time as shown in Figure 4.20.

Instead of 10 seconds, default buffering time of M seconds can be used in case the EAPMST algorithm cannot find a temporal path toward the destination within the next M seconds. In such cases, the buffering time increases with large values of M, thereby, increasing the end-to-end delay as visible from Figure 4.21. In other words, the results from Figures 4.19 and 4.21 show that an increase in the prediction time-step (M) reduces the prediction accuracy and increases the buffering time. That is, the accuracy of prediction algorithm needs to be considered as an important factor while selecting the default

buffering time in SD-DTNs.

From Figures 4.16, 4.20 and 4.21, it is clear that the proposed EAPMST algorithm has a significant role in reducing the packet buffering time. However, the estimation of temporal path in EAPMST depends on the predicted graph sequence $\mathcal{S}'_t$ from Algorithm 4.1. Therefore, the correctness of temporal path depends on how accurate the predicted graph sequence $\mathcal{S}'_t$ will be, i.e., a more accurate sequence $\mathcal{S}'_t$ results in correct temporal path and reduces the buffering time at intermediate nodes. For instance, prediction accuracy of 99.6% in stationary model results in near-zero buffering time at the nodes (see Figure 4.16). On the other hand, a least accurate graph sequence may force EAPMST algorithm to compute less accurate temporal paths, thereby, resulting in longer than desirable buffering time along the flows' path. Our Markov chain-based prediction model is able to provide an accuracy beyond 70% in all mobility models even for 100 future time-steps. However, we limit the number of time-steps (M) to 50 to achieve better performance.

One of the major objectives of SD-DTN framework is to provide network administrators an option to buffer packets in an SDN-controlled manner for addressing link disruptions in SD-WNs. Since the behavior of wireless networks, especially the nature of link disruptions, vary in wide range, network engineers need to design custom network control algorithms depending on the application-specific requirements. For example, the nature of buffering in a software defined vehicular network is entirely different from a software defined WLAN or a software defined satellite network. In this context, our SD-DTN provides an application-agnostic platform for buffering packets depending on the user or business needs. In fact, advanced machine learning techniques such as Recurrent Neural Networks (RNNs) can be employed as SDN control applications in order to accurately predict the future network states and enhance the network performance.

4.6 General Observations

The primary objective of the proposed SD-DTN framework is to provide a buffering-based network control for software defined wireless networks experiencing link disruptions. The experimental results revealed the efficacy of our framework in buffering the packets during link disruptions and forward them as soon as the path toward destination is available for the network flows. In short, we observe the following from our experiments:

1. The newly introduced action STORE performs as expected and forms an effective tool for the controlled buffering of packets.

2. SD-DTN suffers from significant delay in setting up the flow-rules due to the in-band control channel. However, the performance of communication sessions can be improved using modified transport layer protocols such as Dual-mode TCP [66].

3. Mere SD-DTN framework may not be sufficient enough to achieve the required network performance. Additional optimization techniques, such as EAPMST, need to be employed along with SD-DTN to improve the performance as justified from the end-to-end delay characteristics.

4. Maximum end-to-end delay achieved by the communication sessions in Figure 4.13 demonstrates the ability of SD-DTN to buffer packets for significant time during network instabilities.

5. Even though SD-DTN incurs certain amount of OpenFlow overhead, results from Figure 4.14 show that intelligent SDN control algorithms, capable of fine-tuning the HardTimeout (or buffering time) values, can significantly reduce the overhead even below the overhead in traditional SDN schemes.

6. The selection of default buffering time, used in the absence of temporal path, should consider the accuracy of graph prediction algorithm.

7. A detailed study is required on the performance of SD-DTN and SD-DTN+EAPMST schemes on very low and very high data-rate applications operating over unreliable transport layer protocols.

8. The proposed SD-DTN framework opens avenues for designing SDN control applications focusing on efficient buffer management strategies considering the buffer space available at the switches while taking the STORE/FORWARD decisions.

4.7 Summary

In this chapter, we proposed the modelling and design of a Software Defined Disruption Tolerant Networking framework for controlling wireless networks that are prone to link disruptions. The chapter started with a temporal graph-based modeling of the disruption-prone wireless networks followed by the design of an algorithm for predicting the future graph sequence. Further, a new Earliest Arrival Path with Minimal Storage Time algorithm was proposed to compute a possible temporal path for the packets toward the destination, thereby, computing the STORE/FORWARD network control decision. An

SD-DTN switch prototype was developed for enabling the controlled buffering capability with a new STORE action. Further, an SD-DTN testbed was realized using an SD-WMN involving nodes following stationary, group, and random mobility patterns. The proposed SD-DTN scheme along with EAPMST was tested with the traditional SDN approach and was analyzed using the six performance metrics. The results demonstrated the efficacy of the proposed SD-DTN framework in buffering packets during link disruptions and forwarding them when appropriate links get established. Since wireless networks form a major component of next generation communication infrastructures, we find several application domains for our SD-DTN framework as discussed in Chapter 5.

Chapter 5

Emerging Areas of Software Defined Disruption Tolerant Networks

"*Nothing have value without being an object of utility.*"

— Karl Marx

Even though the idea of Delay/Disruption Tolerant Networking (DTN) traces its roots in interplanetary Internets [81], the concept becomes prominent in terrestrial wireless network environments which are prone to link disruptions such as wireless LANs, vehicular networks, tactical networks, and emergency response networks. Since such classes of wireless networks are characterized by uncertain mobility patterns and real-time QoS requirements, controlling the network with traditional algorithms may not be sufficient to achieve the requirements of the present day wireless networks. In this context, SDN's flexible network control offers a promising option to execute network control algorithms based on real-time granular requirements. Moreover, the idea of controlled buffering can play a crucial role in tolerating link disruptions with the help of predicted network states. In this chapter, we discuss some of the disruption-prone wireless domains, along with our contributions, which can be benefited from the proposed Software Defined Disruption Tolerant Networking approaches. In this chapter, we consider the following types of SD-DTNs: (*i*) software defined vehicular networks and (*ii*) software defined satellite networks.

5.1 Software Defined Vehicular Networks

Vehicular networks form one of the important tools of information dissemination in future communication infrastructures. Besides a carrier of data to remote villages which do not

have direct communication links to the outside world as discussed in [138], vehicles play a crucial role in the design of smart urban environments and intelligent transportation systems, especially as a real-time data gathering and communication tool [139–142]. Modern vehicles are equipped with advanced sensor-based electronics to collect and communicate the environmental parameters such as temperature, humidity, and air quality apart from the road traffic related data including the speed of vehicles, density of vehicles, and traffic congestion at the junctions. Different networking techniques are adopted to communicate the vehicles' sensed data to the intended destination depending on the nature of vehicular environment.

Vehicular networks primarily use two communication modes for transferring the data toward the destination: Vehicle-to-Infrastructure (V2I) and Vehicle-to-Vehicle (V2V). In V2I mode, the node residing in the vehicles, either the end-user carrying the mobile phone or the vehicle-mounted network device, communicates directly to the base stations deployed at the roadsides. In V2I, the vehicles do not communicate between each other while transferring the data. On the other hand, V2V enables vehicles to communicate between each other such that the data can be routed to the destination by exploiting the mobility of vehicles in addition to the wireless communication medium. In fact, the advancements in communication technologies including 5G, SDN, and Network Function Virtualization (NFV) lead to the next generation of vehicular infrastructures referred as Vehicle-to-Everything (V2X) and Internet of Vehicles (IoV) [143, 144].

Even though V2X and IoV offer promising solutions for the future communication requirements, the limitations of communication hardware as well as the mobility of vehicles make their realization a challenging task. The communication hardware equipped within the end-user terminals (such as mobile phones, tablets, and laptops) and vehicle-mounted devices offer limited communication range and, therefore, users cannot communicate with the Base Stations (BSes) or Access Points (APs) beyond a distance during their move. In addition, mobility makes the vehicles remain under a single BS for a small amount of time, thereby, necessitating frequent handoffs to the upcoming BSes or APs. However, the BSes and APs are sparsely distributed due to their high cost of deployment and may not be sufficient in number to offer seamless connectivity.

The problem of sparse BS deployment is more severe in V2X and IoV models especially if the hardware is integrated with 5G technology involving advanced networking techniques such as SDN and NFV. In such cases, the BSes play the role of SDN switches or controllers in addition to the role of traditional cellular communication access points. In order to address the long-term absence of network coverage due to sparse BS distribution,

Roadside Units (RSUs) are placed along the roads so that vehicles can get an intermediate access point to the external network besides the existing BSes. In Software Defined Vehicular Networks (SD-VNs), RSUs play the role of SDN switches as well as SDN controllers. In case of RSUs equipped with switches as proposed in [145], they acts as relay points for the data to/from end-users reside within the vehicles as the case with V2I communication mode. On the other hand, RSUs are integrated with SDN controllers to provide flow-rules to the vehicles which are equipped with SDN-switches as in V2V infrastructures [146]. In both cases, the network is prone to frequent link disruptions in control channel as well as in data plane. In this context, our self-configuration scheme offers a suitable solution for handling link disruptions in control channel and managing vehicular-node handoffs to appropriate RSU-based controllers. Along with the self-configuration schemes, the proposed SD-DTN framework can handle link disruptions in data plane and make the vehicles to buffer packets until suitable RSU/vehicle comes into contact.

In both V2I and V2V mode, it is important for the vehicles to frequently communicate with the RSUs either for data dissemination or for achieving network control in dynamic vehicular environments. That is, RSUs need to be placed at appropriate locations in such a way to maximize the vehicle-RSU contacts. However, the placement of RSUs remains an open challenge due to the wide scope of using RSUs in vehicular networks. In this thesis, we propose an RSU placement scheme for SD-VNs with the aim of improving the number of vehicle-RSU contacts, thereby, providing maximum switch-controller interaction. We exploit the structural properties of road networks such as road-length, road-type, and the importance of each junction to define a new metric called *Effort* and deduce an *Effort-Graph* from the road network. Further, the concept of betweenness centrality [137] is applied on the *Effort-Graph* to compute the RSU locations.

5.1.1 *Effort*: A Novel Metric for Road Networks

The three important factors that influence the path of any journey include the path-length between the source and destination points, the possible obstructions in the path such as traffic jams at the intermediate road junctions, and the types of roads the users prefer/need to travel along the path. The path-length represents the geographic distance between the source and destination along the path while the type of a road indicates quality of that road in terms of the width, number of lanes, speed limit, and the number of road intersections. In fact, the type of a road has significant influence in reducing the travel time and providing user comfort. The traffic congestion at a road intersection is primarily

due to the importance of that junction in the overall traffic of the road network.

Figure 5.1 shows the road network of the city of Bangalore, India, where the distribution of colors represents the distribution of road-types. We considered 25 road-types from the OpenStreetMap highway specification [147] and divide them into 12 classes as provided in Table 5.1. Roads with the least type-value offer the highest quality while the lowest quality roads possess the highest type-value. In our method, *motorways* with type-value 1 offers the highest quality road while *footways, steps,* and *paths* offer the lowest quality, i.e., with type-value 12. For designing the metric *Effort*, we consider road-type as

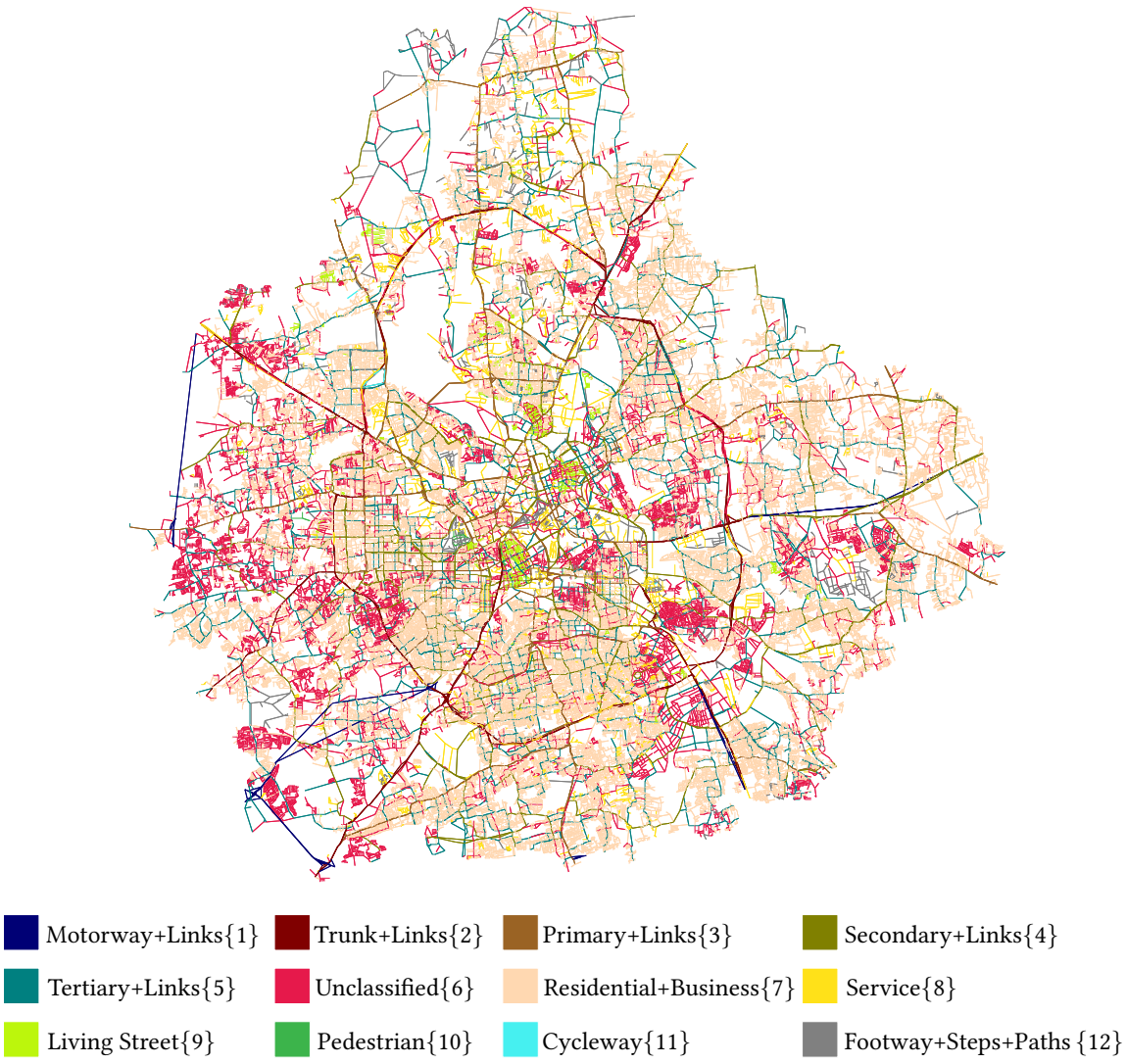


Figure 5.1: Road network of the city of Bangalore, India. The color-code represents 12 classes of road types we considered for our analysis. The type-value of each road-type is provided in braces.

Table 5.1: Classification of road-types. The road-types with value 1 represent the most important roads while the types with value 12 represent the least important roads.

Road-type value	Road-type(s) specified in OpenStreetMap
1	*Motorway* and *motorway_link*
2	*Trunk* and *trunk_link*
3	*Primary* and *primary_link*
4	*Secondary* and *secondary_link*
5	*Tertiary* and *tertiary_link*
6	*Unclassified*
7	*Residential/business*
8	*Service*
9	*Living_street*
10	*Pedestrian*
11	*Cycleway*
12	*Track, footway, bridleway, steps,* and *path*

an important parameter, as suggested in [148], along with the road-length and importance of junctions.

In order to compute the positions of RSUs, we consider the road network as an undirected graph $G = (V, E)$ where V is the set road junctions (nodes or vertices) and E represents the set of roads (links or edges) connecting the junctions. Each edge $(u, v) \in E$ is associated with two attributes: (*i*) $l_{u,v} \in \mathbb{R}^+$ representing the length of the road segment and (*ii*) $t_{u,v} \in \{1, 2, \ldots, 12\}$ representing the type-value of the road. In order to capture the difficulty incurred by users at the junctions, we make use of the metric Betweenness Centrality (BC) defined in [137]. BC of a node measures that node's importance in the shortest paths between other nodes in the network. In a graph, the BC of a node u is computed as

$$C_B(u) = \sum_{i \in V} \sum_{\substack{j \in V \\ i \neq j}} \frac{g_{ij}(u)}{g_{ij}}, \tag{5.1}$$

where g_{ij} represents the number of shortest paths between nodes i and j and $g_{ij}(u)$ is the number of shortest paths pass through node u. We combine the three above-mentioned structural characteristics to define a new metric *Effort* required for a user to travel from junction u to junction v through the link (u, v) as

$$\mathcal{F}_{u,v} = l_{u,v} \times t_{u,v} \times (1 + C_B(v)). \tag{5.2}$$

For computing *Effort* for a link (u, v), we assume that the user just passed the junction u and moves toward junction v incurring the effort due to length $l_{u,v}$, quality of the road $t_{u,v}$, and the difficulty at the junction v, i.e., $C_B(v)$. Moreover, the metric *Effort* provides directionality (i.e., $\mathcal{F}_{u,v} \neq \mathcal{F}_{v,u}$) for each link and subsequently for the source-destination path which cannot be captured using the metrics length and road-type alone.

5.1.2 *Effort*-based Betweenness Centrality for RSU Placement

Since *Effort* captures the real-world characteristics of users' path selection for a journey, the metric can be used for placing RSUs on the most preferred or the least-effort paths in order to achieve maximum vehicle-RSU contacts. Therefore, we deduce an *Effort-graph* $G^{\mathcal{F}} = (V, E^{\mathcal{F}})$ from the original road network and compute the RSU positions depending on the centrality of junctions in $G^{\mathcal{F}}$. In *Effort-graph* $G^{\mathcal{F}}$, each link $(u, v) \in G$ is replaced with two directional links (u, v) and (v, u) weighted with their corresponding *Efforts* as shown in Figure 5.2. An example graph G and its corresponding *Effort-graph* $G^{\mathcal{F}}$ are shown in Figure 5.3.

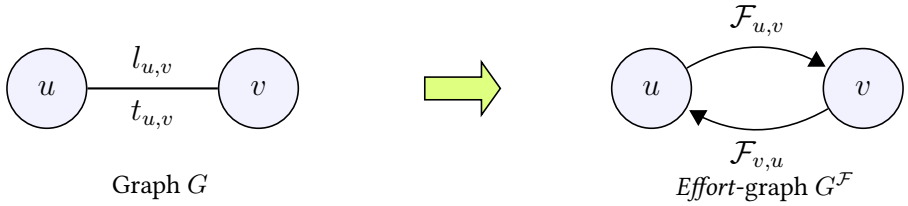


Figure 5.2: Translation of an edge in G to corresponding edges in $G^{\mathcal{F}}$.

Upon generating the *Effort-graph*, we use the BC measure on $G^{\mathcal{F}}$ to compute the RSU positions. Since $G^{\mathcal{F}}$ is directed, BCs of junctions are computed based on the directed shortest paths between node-pairs in $G^{\mathcal{F}}$ and the RSUs are placed on the road junctions of highest BCs. In short, our RSU placement problem is formulated as follows: *Given a road network spanning an area of dimension $A \times A$, select N road junctions for placing RSUs in such a way to maximize the number of vehicle-RSU contacts.*

It is observed that the junctions possessing highest BCs concentrate near the center of the city, thereby, preventing the vehicles at the countryside from accessing the RSUs. Therefore, we consider the given city area as $\sqrt{N}$ square blocks and ensure that each block is equipped with at least one RSU, i.e., RSUs are placed with a minimum inhibition distance μ between them. However, the structure of road networks may not provide sufficient road junctions to place RSUs with $\mu = \frac{A}{\sqrt{N}}$. Therefore, we set $\mu = \frac{A}{2\sqrt{N}}$. The procedure for

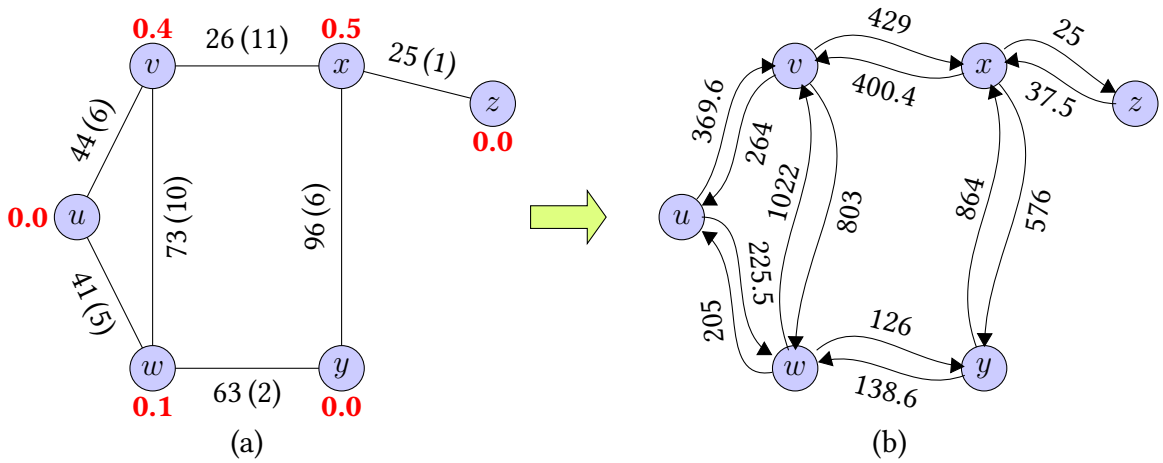


Figure 5.3: (a) An example graph G with six junctions. The length of each link is notated beside the link along with the road-type specified within brackets. The BC value of each junction is provided near the junction in bold face. (b) *Effort-graph* with links weighted with their corresponding *Effort*.

Algorithm 5.1: Procedure for the selection of RSU positions.

Data:
N — Number of locations to place RSUs
$\mathcal{L}$ — List of junctions arranged in the descending order of their BCs in $G^{\mathcal{F}}$
$d_{u,v}$ — Geodesic between vertices u and v
μ — RSU inhibition distance

$\mathcal{R} \leftarrow \{\}$ ▷ Set of selected RSU locations
foreach $u \in \mathcal{L}$ **do**
  ValidPosition $\leftarrow$ *True*
  foreach $v \in \mathcal{R}$ **do** ▷ Check μ with already selected junctions
    if $d_{u,v} \leq \mu$ **then**
      ValidPosition $\leftarrow$ *False*
      Break
    end
  end
  if ValidPosition $=$ *True* **then**
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{u\}$
  end
  if $|\mathcal{R}| = N$ **then**
    return $\mathcal{R}$
  end
end

selecting the RSU positions is described in Algorithm 5.1. The algorithm starts selecting the nodes (junctions) from the list $\mathcal{L}$ from the first node onward (Line 2). For each selected node, the algorithm ensures that the node is at least inhibition distance μ away from the already selected nodes (Lines 4–9). Since the length of list $\mathcal{L}$ is $|V|$, the worst-case time complexity of the algorithm is estimated to be $\mathcal{O}(N|V|)$.

5.1.3 Performance Analysis

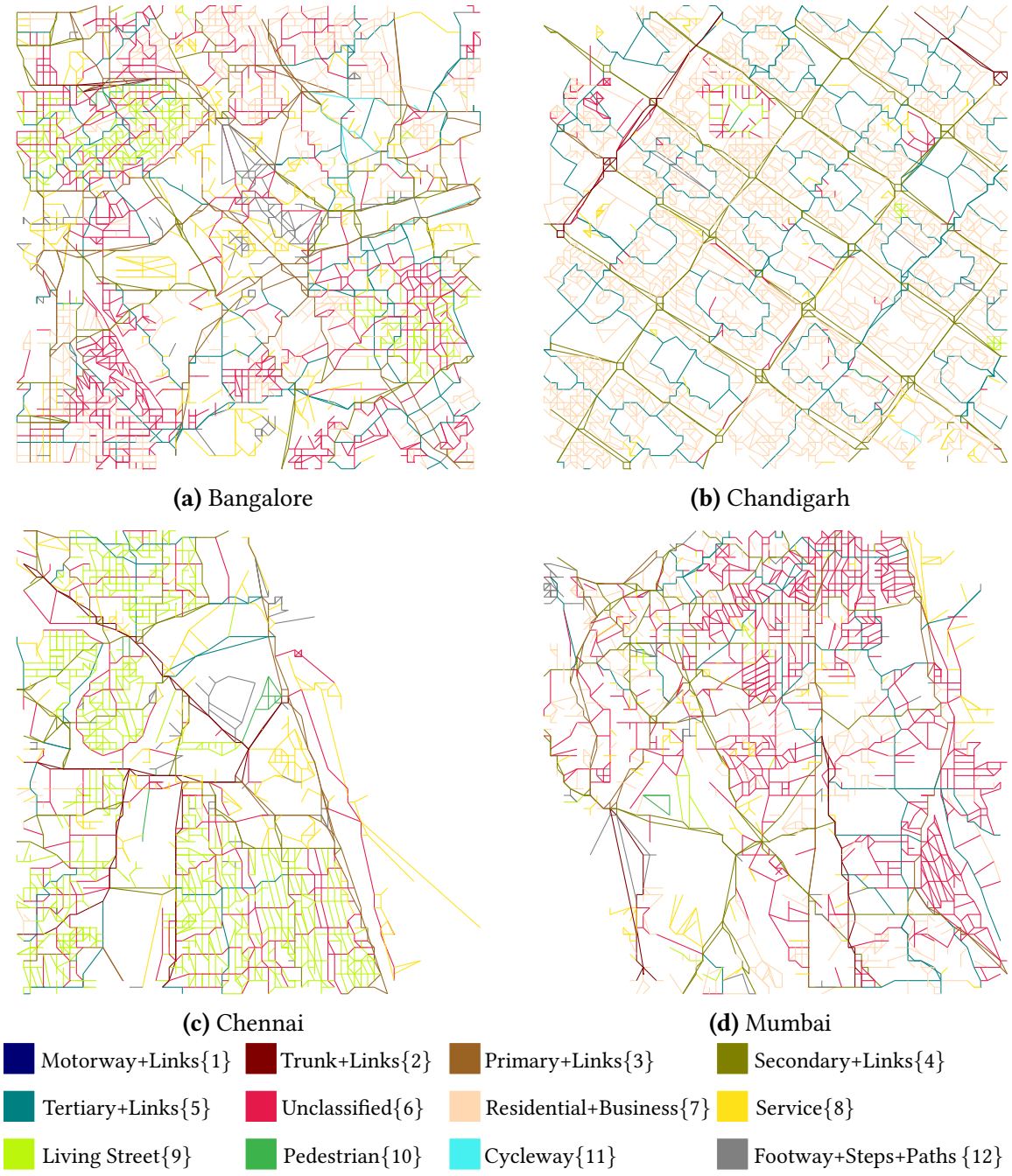
In order to analyze the performance of the proposed *Effort*-based Betweenness Centrality (EBC) scheme, we consider the following four city road networks from India: (*i*) Bangalore, (*ii*) Chandigarh, (*iii*) Chennai, and (*iv*) Mumbai. The road networks of the four cities considered are shown in Figure 5.4. Except Chandigarh, the cities are characterized by complex road networks and are vulnerable to traffic congestion. On the other hand, Chandigarh is a planned city and the road network forms a grid structure as can be seen from Figure 5.4(b). In order to generate different traffic patterns, we used the Simulation of Urban MObility (SUMO) [149] simulator with random trips, whose source and destination are selected at random, generated at every second. For analyzing the performance, the following three traffic patterns are used:

1. **Shortest path based on road-length:** Here, the user computes the shortest path from source to destination using the length of the roads and *does not* consider the type of the road. However, travelling along such a path may not be always feasible in real-world due to the presence of low quality roads. For example, a user travelling in a car cannot go through a shortest path involving footways.

2. **Shortest path based on user priority:** In order to overcome the drawback of ideal length-based shortest paths, we define a more realistic traffic pattern based on users' priority of road-types. Here, users select their minimum preferred road-type t_p for a journey and travel through only the roads of type t_p or lower. Therefore, the road network gets restricted to a preference graph $G_p = (V, E_p)$ where $E_p = \{(u, v) \in E \wedge t_{u,v} \leq t_p\}$ and the shortest path for the journey is computed over G_p.

3. **Shortest path based on travel time:** Here, users consider the length of the road, width of the road, number of lanes, and the speed limit on each road for minimizing the travel time. The traffic generation algorithm, called *Duarouter* [150], integrated with the SUMO simulator considers the above-mentioned factors along with the

possibility of alternate paths for the journey depending on the dynamic traffic conditions. *Duarouter* provides the most accurate real-world traffic pattern of users' journeys among the three traffic patterns under consideration.



(a) Bangalore **(b)** Chandigarh

(c) Chennai **(d)** Mumbai

Figure 5.4: City road networks from India with dimension $5\,\mathrm{km} \times 5\,\mathrm{km}$ around the city center. The color code represents 12 classes of road-types and the type-value of each road-type is given within braces.

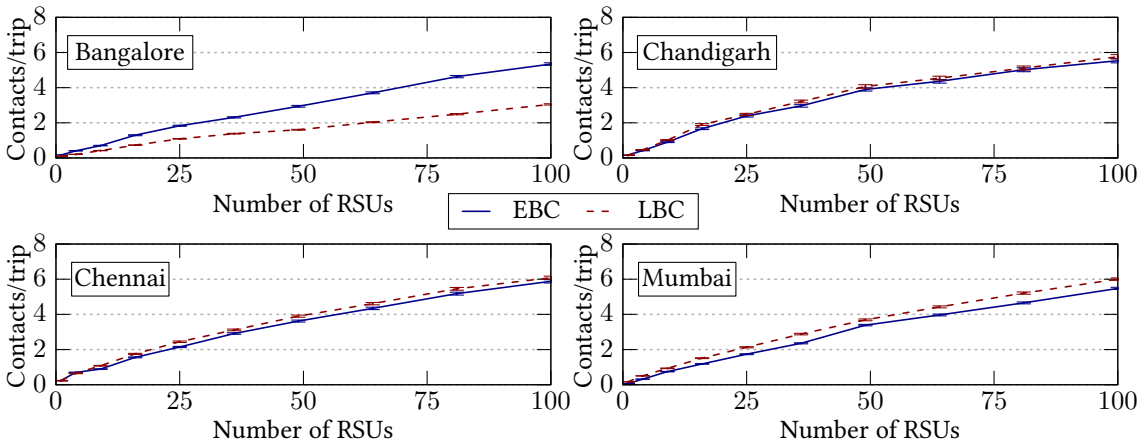
Table 5.2: Simulation parameters for RSU placement schemes.

Parameter	Value/Description
City maps	Bangalore, Chandigarh, Chennai, and Mumbai
Map source	OpenStreetMap
City area	$5\,\mathrm{km} \times 5\,\mathrm{km}$
Road-types	12 (as provided in Table 5.1)
Vehicle speed	As per the OpenStreetMap specification [152]
Communication range	100 meters
Simulators	SUMO [149] and UDTNSim [153]
Number of road-traffic samples	50

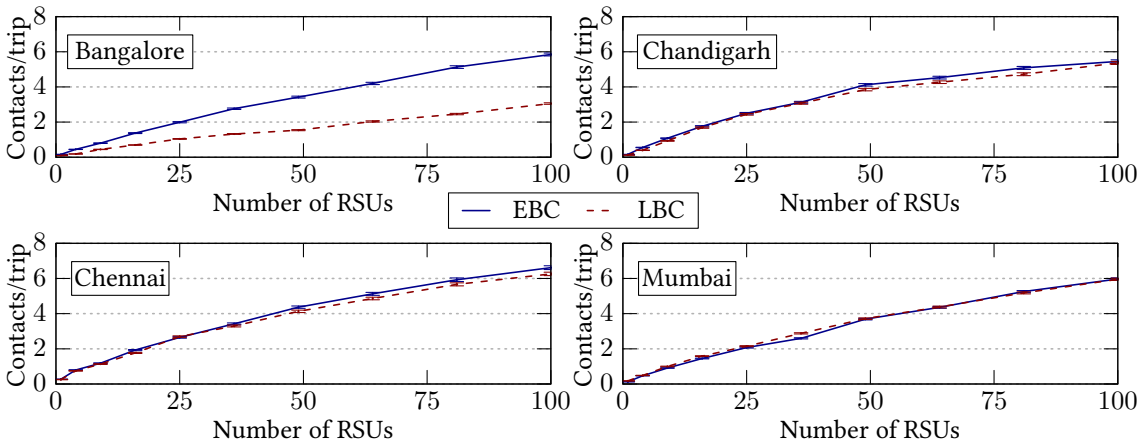
The simulation parameters we considered are provided in Table 5.2. The city road networks within an area of dimension $5\,\mathrm{km} \times 5\,\mathrm{km}$ around the city center are extracted from OpenStreetMap and the communication range of vehicles and RSUs is assumed to be 100 meters. Due to the random nature of traffic patterns, we generated 50 samples for each type of the three traffic patterns discussed above and the performance is analyzed using the average values computed from the 50 samples. The measures used to analyze the performance include: (*i*) *contacts per trip*, (*ii*) *contact probability*, and (*iii*) *contact time*. The performance of EBC scheme is compared with the *Length-based Betweenness Centrality* (LBC) scheme proposed in [151] since it provides the best performance among the other RSU placement schemes. In LBC scheme, the BCs of nodes are computed on the original graph G with road-length as the edge-weight. Further, N nodes offering the highest BCs are chosen for the RSU placement keeping the inhibition distance of μ. In other words, LBC follows the same procedure as Algorithm 5.1 except that the list $\mathcal{L}$ represents the nodes arranged in the descending order of BCs on the original road network G.

5.1.3.1 Contacts per Trip

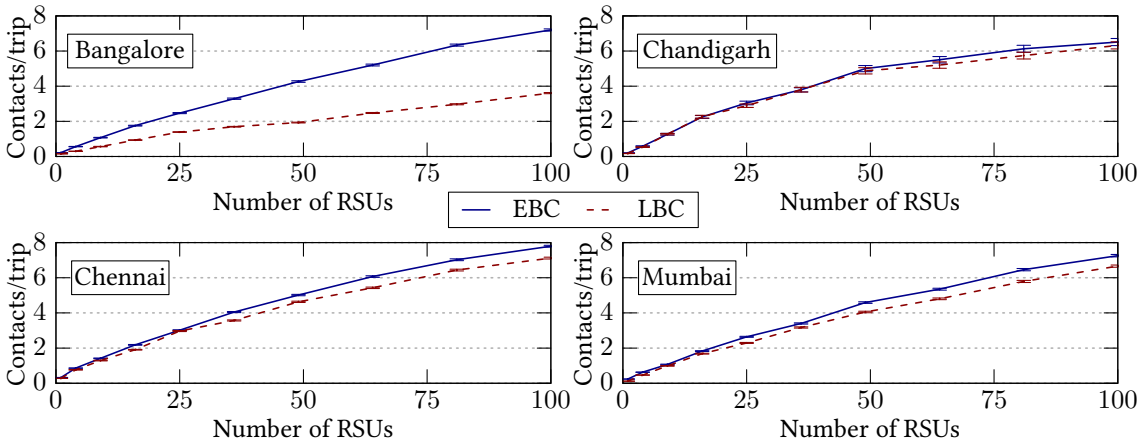
The primary objective of our RSU placement problem is to maximize the number of vehicle-RSU contacts. We define contacts per trip as the average number of times a vehicle comes in contact with RSUs in a trip. In SD-VNs running V2I mode, it is important for a vehicle to improve the number of RSU contacts in order to offload the data generated from the vehicle or to forward the data destined to the vehicle. On the other hand, in SD-VNs running V2V mode, the vehicles playing the role of SDN switches need to frequently contact RSUs to obtain the flow-rules from controllers residing at the RSUs. Figure 5.5 shows the average number of vehicle-RSU contacts obtained under the three traffic scenarios



(a) Shortest path based on road-length.



(b) Shortest path based on user priority.



(c) Shortest path based on travel time.

Figure 5.5: Contacts per trip under different traffic patterns.

considered. In the ideal shortest path-based traffic as in Figure 5.5(a), our EBC approach does not perform as good as the traditional LBC scheme. As we introduce a more realistic traffic scenario based on the user-preferred road-types, we can observe that the proposed EBC scheme approaches the LBC curve and starts dominating over the LBC scheme as shown in Figure 5.5(b). Unlike the above two traffic patterns, our approach offers a clear advantage over the LBC scheme in dynamic real-world traffic scenarios as revealed from Figure 5.5(c). Such an increase in the number of vehicle-RSU contacts is important in SD-VNs especially when RSUs play the role of SDN controllers. Besides contacts per trip, it is important to verify whether the EBC scheme has any influence over other performance metrics as discussed in the next two subsections.

5.1.3.2 Contact Probability

Contact probability measures the chance that a user meets with at least one RSU in a trip. We compute contact probability as the ratio of the number trips having at least one RSU contact to the total number of trips. From Figure 5.6 we can observe that in all city road networks considered, our EBC approach provides same or better probability for meeting with an RSU in a trip. The probability curves show that with increase in the number of RSUs N, the contact probability increases. Also, it is important to note that with a city area of dimension $5\,\mathrm{km} \times 5\,\mathrm{km}$, 50 RSUs are sufficient with EBC to provide contact probability beyond 0.9.

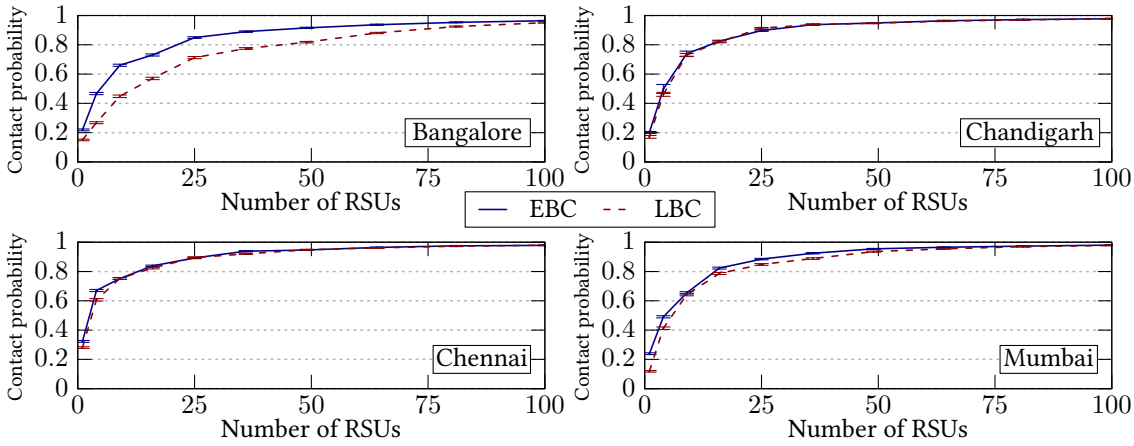


Figure 5.6: Contact probability under the shortest path traffic based on minimum travel time.

5.1.3.3 Contact Time

Contact time is defined as the difference between the time a vehicle enters the communication range of an RSU and the vehicle leaves the communication range. In other words, the average time a vehicle remains with an RSU in a contact. In terms of V2I mode, where the RSUs act as SDN switches, the vehicles require significant contact time either to offload the data packets generated from the vehicles or to deliver the packets residing at the RSUs to the vehicles. On the other hand, small contact time is sufficient for RSUs (controllers) in V2V mode for providing the flow-rules to the vehicles (SDN switches). Figure 5.7 shows the average contact time offered by the EBC and LBC schemes under real-world traffic scenarios based on minimum travel time. We cannot observe a significant change in contact time with varying number of RSUs. However, the contact time lies between 10 and 20 seconds in all cities except Chandigarh, where the contact time reaches 40 seconds. Such a high contact time in Chandigarh may be due to the grid structure of road networks as the case with any planned city. In fact, the contact time can be influenced by other factors such as the speed of vehicles and the angles the roads make with a junction.

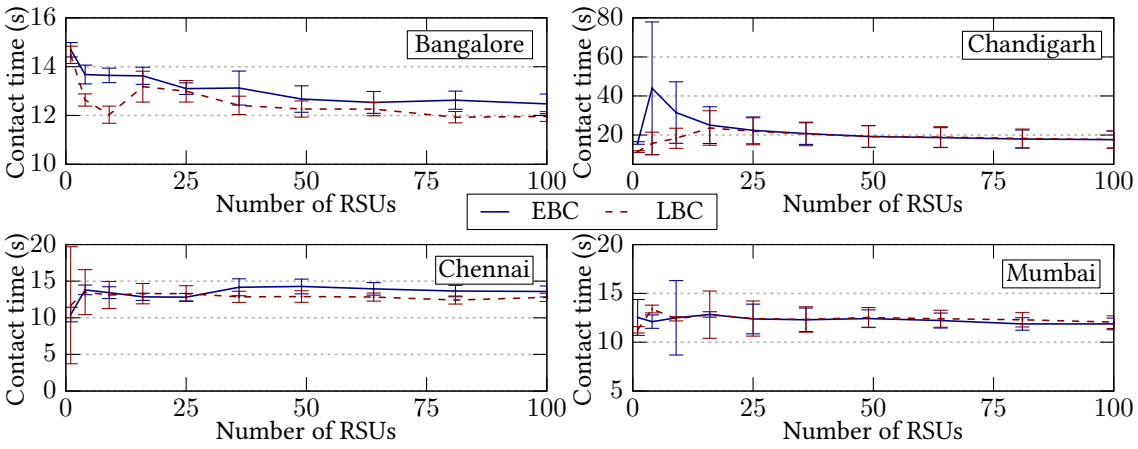


Figure 5.7: Contact time under the shortest path traffic based on minimum travel time.

5.1.3.4 Delivery Ratio

In order to test the efficacy of EBC-based RSU placement scheme in a real-world-like environment, we simulate a data gathering scenario using vehicles equipped with sensors and measure the delivery ratio with respect to the traditional LBC scheme. We used Urban Delay Tolerant Network Simulator (UDTNSim) [153], with input as the vehicular traffic

trace from SUMO, to simulate the data gathering from the city of Bangalore, India, using V2I mode as well as V2V mode. Here, RSUs act as SDN switches (both in V2I and V2V) to relay the data from vehicle-mounted sensors and there does not exist direct communication between the RSUs, i.e., the RSUs communicate *only* via vehicles. The Command and Control Center (CC) is assumed to be placed at the city center, i.e., the junction which provides the highest BC. Sensors in each vehicle generate data in every minute and send to the CC via multihop path using either V2I or V2V mode. The delivery ratio is computed as the ratio of the number of messages delivered to the CC to the total number of messages generated at the vehicle-mounted sensors.

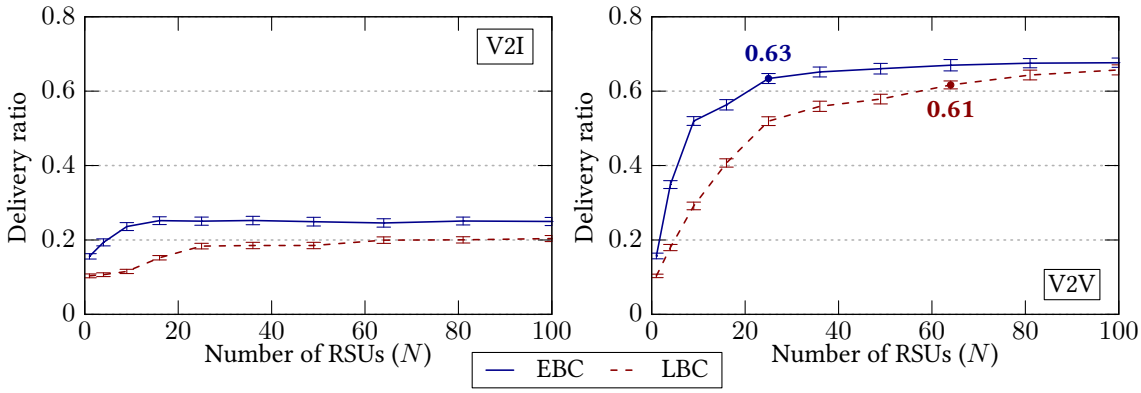


Figure 5.8: Delivery ratio offered by the EBC and LBC schemes in a data gathering scenario.

Figure 5.8 shows the delivery ratio offered by the EBC- and LBC-based RSU placement schemes in V2I and V2V mode of communication. In both cases, our proposed EBC scheme provides higher delivery ratio (with $> 30\%$ increase) than the LBC scheme for all number of RSUs. As expected, the delivery ratio is higher in V2V mode compared to V2I mode. Moreover, the EBC-based scheme requires only 25 RSUs to achieve delivery ratio of 0.6 while LBC requires nearly 60 RSUs. The delivery ratio beyond 0.7 in V2V mode indicates the scope of improving the data forwarding operations with intelligent techniques, thereby, meeting the demands of future communication systems.

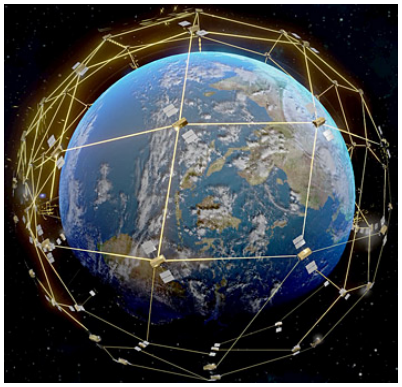
Software defined vehicular networks play a major role in future communication infrastructures in terms of providing Internet when people are on their move. Due to the frequent link disruptions resulting from the limited communication range and vehicle mobility, it is important to place the RSUs at appropriate locations minimizing the chance of link disruptions, thereby, offering seamless connectivity. Therefore, our RSU placement scheme along with the self-configuration and SD-DTN framework can be a best-fit can-

didate for providing low-latency communication services in future SD-VNs depending on the user mobility pattern and network dynamics.

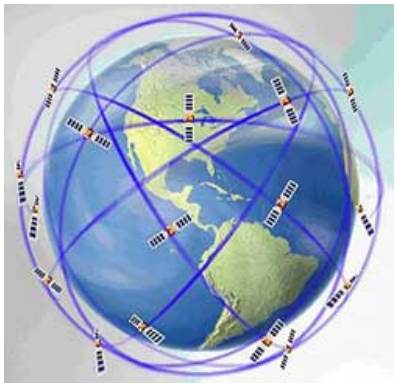
5.2 Software Defined Satellite Networks

Artificial satellites constitute one of the major components of modern communication systems, especially in remote areas where the limited infrastructure and network coverage form the primary concerns. In space-based communication systems, satellites act as repeaters in space and relays the data to a much larger area compared to the coverage offered by the existing terrestrial infrastructure. The satellites dedicated for broadcasting purposes are placed in Geosynchronous Equatorial Orbits (GEO), at an altitude of $35{,}786\,\mathrm{km}$ from the Earth's surface, to keep them stationary with respect to the sender and the receiver. However, the long propagation delay in the order of hundreds of milliseconds over such high altitudes makes geostationary satellites not appropriate for meeting the modern day personal communication requirements. For example, 5G cellular networks are designed with the objective of reducing the end-to-end latency below 1 millisecond [75]. Therefore, Low Earth Orbits (LEOs) with altitudes below $2{,}000\,\mathrm{km}$ and propagation delay less than 10 milliseconds are recommended to deploy the satellites destined for low-latency communication systems. The satellites residing at LEOs require low transmission power, which further reduces the energy requirements, complexity of electronic circuitry, and consequently, the size and cost of the satellites. On the other hand, satellites in LEOs have small orbital periods in the order of 1–2 hours which make them unavailable to a specific ground point for the majority of time. However, such an absence can be nullified by deploying multiple satellites, known as a *satellite constellation*, in such a way to provide uninterrupted communication, i.e., at least one satellite is made available for communication at any point of time on the area of interest (see Figure 5.9). The earliest examples of satellite constellations include Iridium [158] involving 66 satellites in LEO polar orbits with an average altitude of $780\,\mathrm{km}$ and Globalstar [159] consisting of 48 satellites in circular LEOs at an altitude of $1{,}389\,\mathrm{km}$. The recently proposed/on-going satellite constellation deployments aiming at providing broadband Internet at a global scale include Starlink by SpaceX [156] with 12,000 satellites (possible extension to 42,000 satellites) at the three shells of LEOs, OneWeb constellation [157] with 720 satellites at LEOs with an altitude around $1{,}200\,\mathrm{km}$, and Telesat [160] with 117 satellites at altitudes 1000–$1200\,\mathrm{km}$ [161].

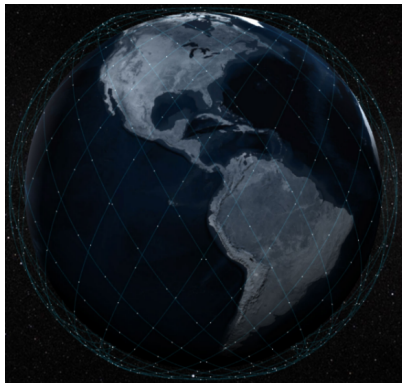
As suggested in [162], satellites can help modern communication systems in the following three different ways: (*i*) as a Layer-2 trunk link between two points, (*ii*) as a subnet,



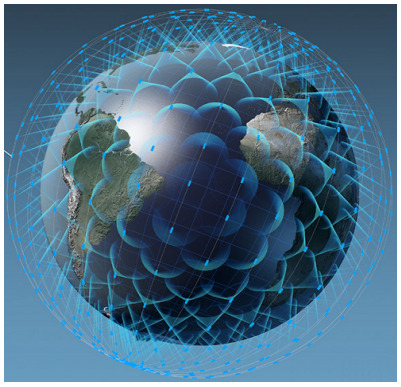
(a) Iridium [154].



(b) Globalstar [155].



(c) Starlink [156].



(d) OneWeb [157].

Figure 5.9: Example existing/proposed satellite constellations.

i.e., satellites serve as the last hop for Internet access, and (*iii*) as the core network. For instance, Telstra's Satellite IP Trunk Service [163] comes under the category of trunk link service. Among the examples mentioned above, Globalstar and OneWeb come under the class of subnets, i.e., satellites serve as the intermediate node between user and the Internet gateway while the satellites in Iridium and Starlink constitute the core network of the entire infrastructure with Inter-Satellite Links (ISLs). Even though ISLs add overhead on the electronic circuitry and the on-board software systems, such satellite-to-satellite links have significant impact in improving the end-to-end throughput as observed in [161].

Since satellites form a wireless backbone in space, Software Defined Satellite Networks (SD-SNs) can play a major role in improving the network performance especially in providing next generation communication services in areas with limited terrestrial infrastructure. However, the link disruptions in satellite networks lead to an SD-DTN scenario in space segment as well as in terrestrial-space network interface. Link disruptions in satellite networks, both in ISLs and satellite-to-ground terminal links, are attributed to dif-

ferent factors such as: (*i*) limited communication range, (*ii*) absence of Line-of-Sight (LoS) resulting from the mobility of satellites as well as users, (*iii*) orbital seam between ascending and descending satellites, (*iv*) the atmospheric path the signal takes to reach the ground terminals, and (*v*) satellite handoffs by the ground terminals. From the perspective of SD-DTN, the control plane can reside at satellites in GEOs [164] or at the ground stations [165]. In both cases, the satellites in LEOs as well as the ground terminals that constitute the data plane incur control channel disruptions which necessitate appropriate controller handoffs. Since it is difficult to maintain a connected topology in a satellite network due to the orbital characteristics and mobility, Controller-Initiated Handoff becomes an infeasible option. On the other hand, Switch-Initiated Handoff with SD-OLSR protocol offers the required handoff service to the switching devices with appropriate tuning options depending on the constellation characteristics.

Similar to control plane, the satellites and ground terminals in data plane form a DTN and the data packets need to be buffered, as indicated in [166], to overcome the challenges resulting from link disruptions. In such a context, our SD-DTN framework can provide an appropriate platform for administrators to execute the network control algorithms involving controlled buffering of packets at specific satellites. The periodicity of satellite mobility in respective orbits removes the requirement of complex prediction algorithms (similar to the Markov model-based prediction discussed in Chapter 4), thereby, improving the accuracy and performance of EAPMST algorithm. In the following subsection, we discuss two SD-DTN case studies of EAPMST on two existing satellite constellation models.

5.2.1 Case Studies of SD-DTN and EAPMST on SD-SNs

For analyzing the feasibility of SD-DTN and EAPMST on SD-SNs, we considered two satellite constellation models: (*i*) Iridium-NEXT and (*ii*) DebriNet. For the simulation, we use the Systems Tool Kit (STK) [167] simulator for creating the satellite network environment with 12 ground stations of Indian Space Research Organization (ISRO) [168] as the user terminals on the Earth. The ground stations are simulated to be located at Bangalore, Hyderabad, Lucknow, Port Blair, Sriharikota, Thiruvananthapuram (India), Troll (Antarctica), Svalbard (Norway), Port Louis (Mauritius), Bear Lake (Russia), Biak (Indonesia), and Gatun Lake (Panama) as shown in Figure 5.10. The simulations are carried out for a duration six hours, and the performance metrics end-to-end delay and buffering time per node are computed using several communication sessions. In each communication ses-

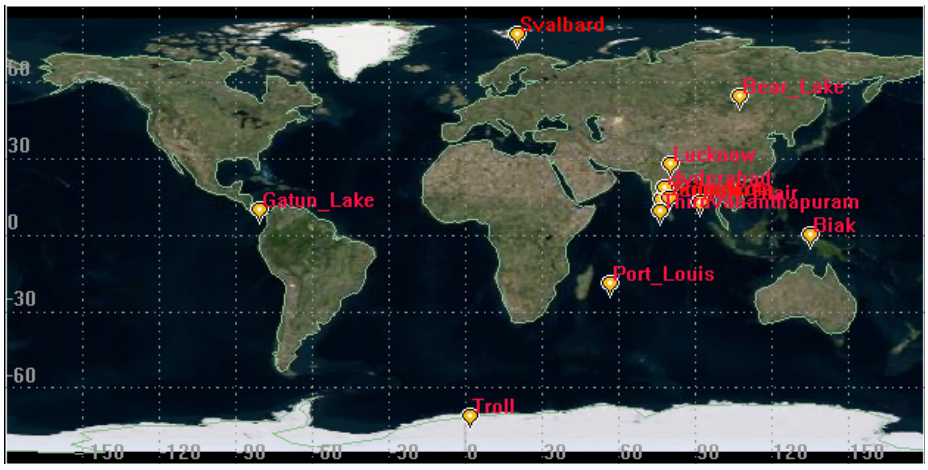


Figure 5.10: Ground stations considered for the analysis of SD-DTN in SD-SNs.

sion, the source and destination nodes are assumed to reside on the Earth and are chosen from the set of ground stations at random. One communication session is generated at every second where the source node sends a message to the destination using the satellite backbone. We assume that a consistent control channel exists between the controller and the switches (satellites and ground stations). In order to compare the performance of EAPMST buffering/routing network control algorithm, we use a DTN routing protocol Earliest-on-Contact (EoC) where a satellite A carrying a message forwards the message to another satellite B during a contact only if B meets with the destination *earlier* than A.

5.2.1.1 EAPMST on *Constrained*-Iridium-NEXT Constellation

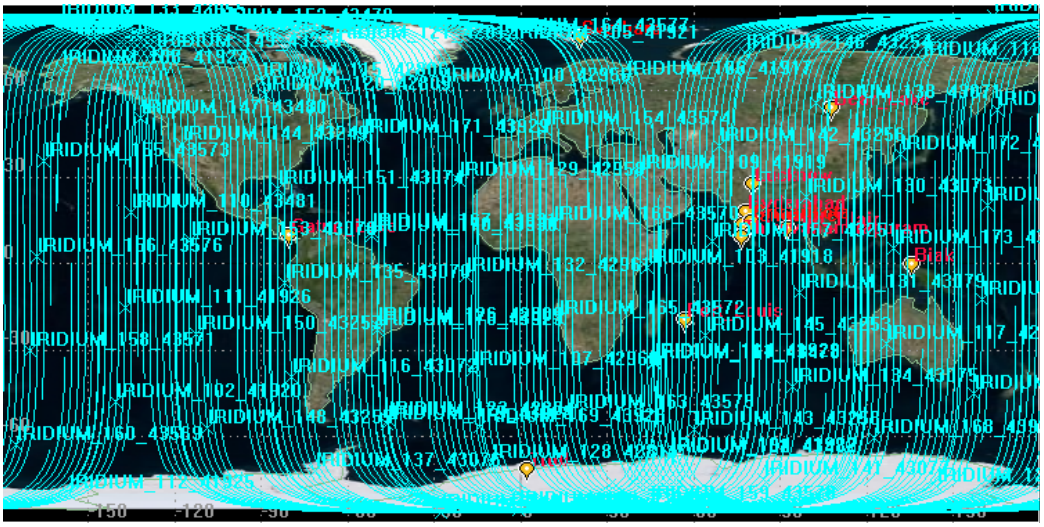
Iridium-NEXT constitutes the next generation of existing Iridium constellation [158]. Besides communication, Iridium-NEXT is equipped with payloads for atmospheric and sea observation. The network consists of 66 active satellites on six orbital planes, 11 satellites on each orbit. The orbits are of polar nature with an altitude of $780\,\mathrm{km}$ at an inclination $86.4°$. The constellation is designed in such a way to provide global coverage at any time with the active satellites, i.e., all points on the Earth comes under the footprint of at least one satellite all the time. Iridium-NEXT also uses inter-satellite links to ensure an end-to-end path between the source and destination points. Each satellite in the constellation is equipped with four ISLs, two with the neighbors in the same orbital plane, one with the satellite residing on the left orbital plane, and another with the satellite on the right orbital plane as shown in Figure 5.11. Further, two users residing on the Earth can communicate with each other at anytime using the satellite backhaul in space. Apart from 66 satellites, nine additional satellites are launched to serve as in-orbit spares to

Figure 5.11: Snapshot of Iridium-NEXT constellation [169].

be utilized during emergency situations such as the failure of satellites. The final set of Iridium-NEXT satellites was launched on 11th January, 2019.

Due to the mesh backhaul and the complete coverage of Earth, there may not exist the need of controlled buffering in Iridium-NEXT constellation unless the network incurs disruptions in the end-to-end paths due to satellite failure or malfunctioning. Therefore, in order to study the efficacy of the proposed SD-DTN in SD-SNs, we assume the satellites and ground stations use an omnidirectional antenna with the communication range limited to $800\,\mathrm{km}$. Details of the satellites we used in the Iridium-NEXT constellation are provided in Table A.1 of Appendix A. We call our experimental constellation as *Constrained*-Iridium-NEXT constellation. Figure 5.12 shows the 3D and 2D snapshots of the *Constrained*-Iridium-NEXT constellation from the STK simulator with 12 ground stations. Such an arrangement results in a DTN scenario on which we execute the EAPMST algorithm for the controlled buffering and routing of packets toward the destination. It is important in satellite networks to consider the duration of a satellite isolation besides the disruption of a specific link between two satellites. In satellite networks, especially where the satellites are distributed in uneven orbits, it is difficult to find an end-to-end path between source and destination at a particular time instant, and the satellites remain *without any link*, i.e., in *isolation*, due to the characteristics of the orbits they move. Therefore, buffering becomes the only option during node isolation and the buffered packets are routed using opportunistic forwarding when an appropriate link gets established with another satellite.

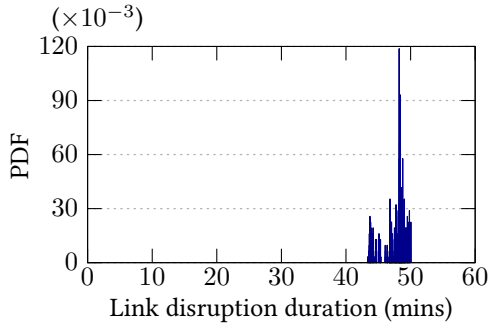
(a) 3D-view of the *Constrained*-Iridium-NEXT constellation.



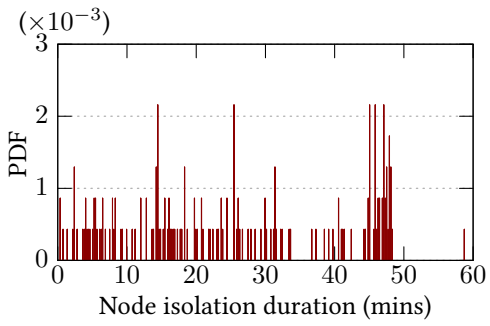
(b) 2D-view of the *Constrained*-Iridium-NEXT constellation.

Figure 5.12: Snapshots of the *Constrained*-Iridium-NEXT constellation from the STK simulator.

Figures 5.13(a) and 5.13(b) show the distribution of the durations of link disruptions and node isolation, respectively, in a scenario with *Constrained*-Iridium-NEXT constellation as the space segment and 12 ground stations residing on the Earth. It is clear from Figure 5.13 that node isolation occurs in a wide range of durations while the link disruptions are concentrated within the range of $40–50$ minutes. A comparison between the



(a) Distribution of link-disruption.



(b) Distribution of node isolation.

Figure 5.13: The distribution of durations of link disruptions and node isolation in *Constrained*-Iridium-NEXT constellation. The distribution is represented using probability density functions.

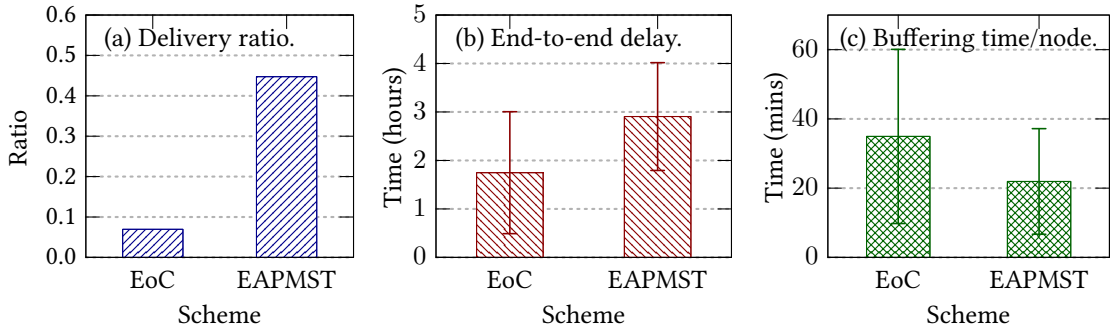


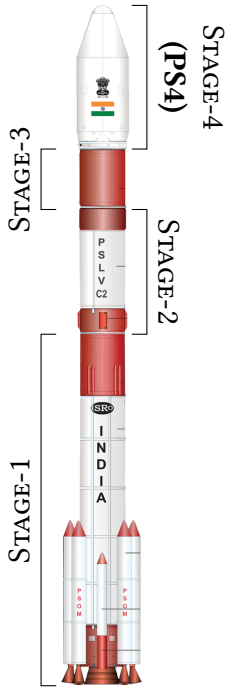
Figure 5.14: Performance of EAPMST on Constrained-Iridium-NEXT constellation.

performances of the EoC algorithm and the proposed EAPMST is shown in Figure 5.14. It is important to note that the proposed EAPMST is able to achieve a delivery ratio of 0.44 while EoC offers only 0.07 within the experiment duration of 6 hours (see Figure 5.14(a)). Due to the lack of an end-to-end path at any time instant, the messages get buffered in satellites for significant time in EoC without getting a contact with an appropriate satellite which can deliver the message to the destination at the earliest. On the other hand, EAPMST makes the messages wait at satellites till specific links get established which can subsequently route the packets toward the destination with minimum buffering time while ensuring earliest delivery. Despite the low delivery ratio, EoC provides an average end-to-end delay lesser than EAPMST as can be seen from Figure 5.14(b). On the other hand, EAPMST is successful in achieving low per-node buffering time of 21.93 minutes than 34.92 minutes in EoC (see Figure 5.14(c)).

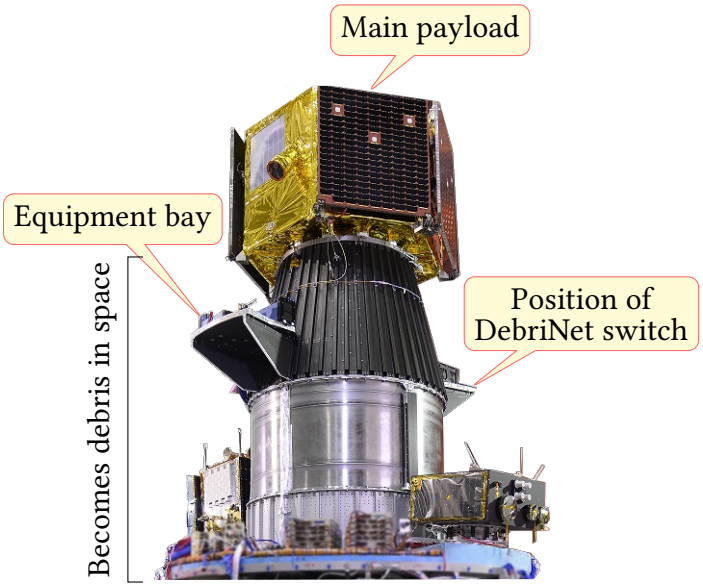
5.2.1.2 EAPMST on DebriNet Constellation

Space debris form a major concern for existing and future space missions due to the harm they cause in terms of possible collisions on the active nodes in space as well as on the launch vehicles, and obstructions in the line-of-sight between different nodes [170, 171]. However, the removal of debris from space remains an open problem among the space research community. As an alternate direction, ISRO has come up with an idea of considering space debris as a platform for carrying out scientific experiments in the field of atmospheric sciences, Earth observation, and communication. Following such a direction, we designed an SDN-based communication architecture called DebriNet in [165] exploiting the space debris to provide a low-cost opportunistic networking platform for common man as well as to realize an Internet of Things (IoT) backhaul in space for the next generation low data-rate applications.

In order to realize DebriNet, we considered the debris of Polar Satellite Launch Vehicles (PSLVs), the workhorse of ISRO in launching satellites. PSLV consists of four stages and during the launch, the first three stages get detached from the vehicle and gets burnt out in the Earth's atmosphere. The fourth stage, known as PSLV Stage 4 (PS4), injects



(a) PSLV stages [172].



(b) Internal structure of PS4 [173]. Upon injecting the main payload, the remaining structure of weight several tonnes revolves around the Earth as a debris for several decades.

Figure 5.15: Stages of PSLV and the internal structure of PS4 module.

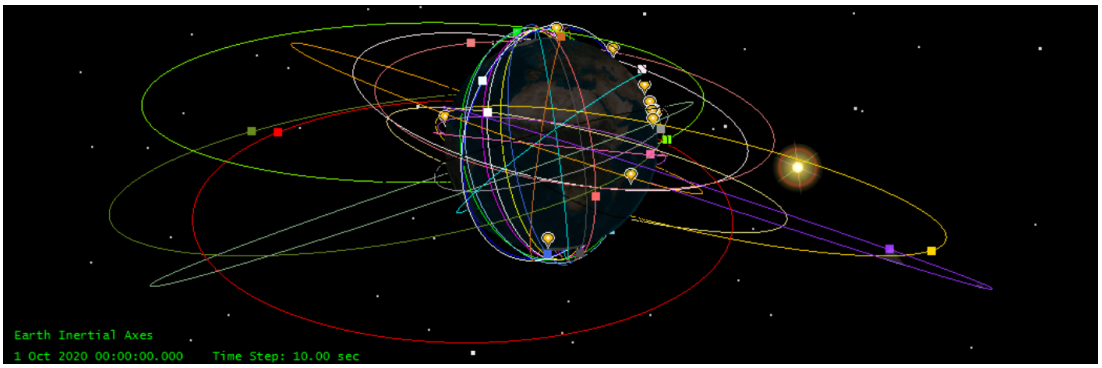
the payload into the required orbit and remains as a debris in space. Figure 5.15(a) shows the four stages of PS4 while Figure 5.15(b) depicts the structure of the PS4 module. An equipment bay is integrated with PS4 to place the flight computer, guidance system, and auxiliary payloads. In addition, PS4s are equipped with solar panels to provide power to different system components. Exploiting the resources available in PS4, DebriNet proposed to deploy an SDN switch at the equipment bay which can act as a networking node in space as long as the solar panels provide sufficient power. The primary advantage of DebriNet is its scalability that each launch can deploy at least one SDN node in space so that the network can provide an end-to-end path between any two points on the Earth in future.

Each node in DebriNet is designed using Commercial Off-The-Shelf (COTS) devices to minimize the cost so that common man can afford to use the network for communication purposes. Therefore, On-Board Computer (OBC) of a DebriNet node is designed using a Raspberry Pi Zero computing board while the communication interface is realized using an advanced LoRa (Long Range) module which uses the frequency of $433\,\mathrm{MHz}$ or $867\,\mathrm{MHz}$. An experiment carried out in Spain [174] showed that LoRa module can provide a communication range of $766\,\mathrm{km}$, which resides within the range of LEOs. The User Equipment (UE) is designed separately using both Arduino and Raspberry Pi boards and integrated with a LoRa module. UEs act as an interface between end-user devices, which use either Bluetooth or WiFi, and the satellites in space which use only LoRa.

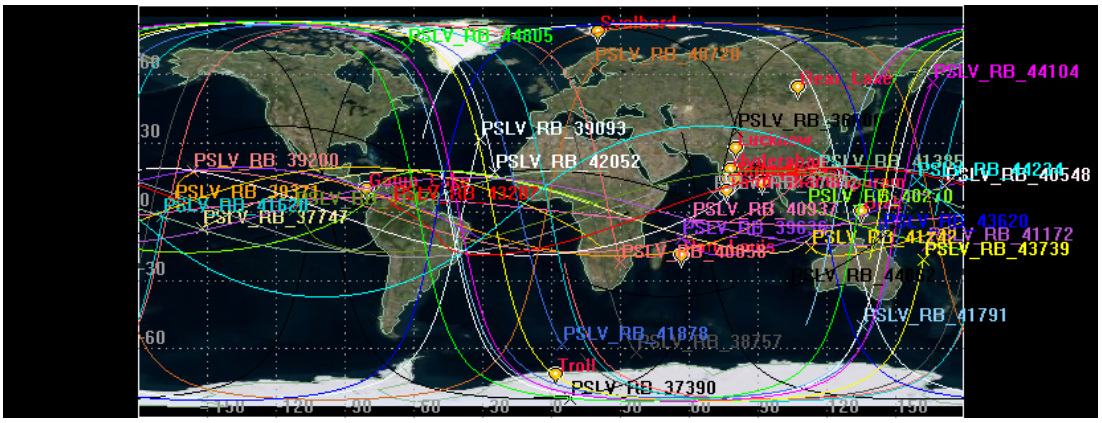
For our simulations in STK, we considered the debris, also known as *rocket bodies*, from 30 PSLV launches from 9^{th} September, 2012 to 11^{th} December, 2019. The details of PSLV rocket bodies we used are provided in Table A.2 of Appendix A. The communication range of debris as well as 12 ground stations is set to $700\,\mathrm{km}$ in order to be within the LoRa interface's transmission range. Figures 5.16(a) and 5.16(b) show the snapshots of DebriNet constellation from 3D and 2D viewports, respectively. Unlike *Constrained*-Iridium-NEXT, nodes in DebriNet revolve the Earth in uneven orbits, which depends on the final injection orbit and the launch manifesto. DebriNet constellation also has node isolation durations distributed in a wide range (Figure 5.17(b)) compared to the link disruption counterpart (Figure 5.17(a)). The performances of EoC and the proposed EAPMST on the DebriNet constellation are provided in Figure 5.18. Similar to *Constrained*-Iridium-NEXT, EAPMST offers delivery ratio of 0.3 which is better than EoC's 0.21. It is important to note that EoC performs better in DebriNet compared to *Constrained*-Iridium-NEXT. Also, EoC offers end-to-end delay lower than EAPMST, however, the difference in delay gets reduced in DebriNet. In terms of the per-node buffering time, the domination of EAPMST is visible

in Figure 5.18(c) similar to that of *Constrained*-Iridium-NEXT.

In order to analyze the scalable nature of DebriNet and subsequent improvement in performance, we considered the performance metrics with increasing number of DebriNet

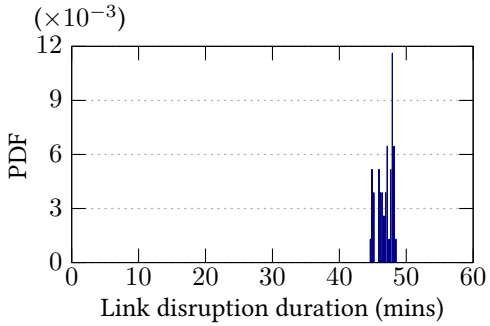


(a) 3D view of DebriNet constellation.

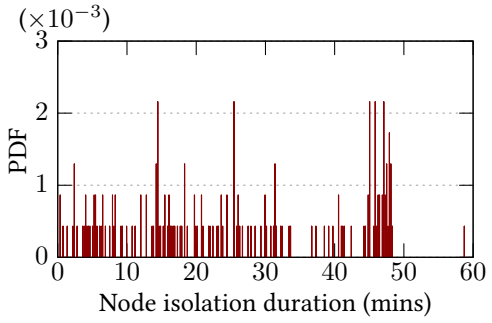


(b) 2D view of DebriNet constellation.

Figure 5.16: Snapshots of DebriNet constellation from STK simulator.



(a) Distribution of link disruption.



(b) Distribution of node isolation.

Figure 5.17: The distribution of durations of link disruptions and node isolation in DebriNet constellation. The distribution is represented using probability density functions.

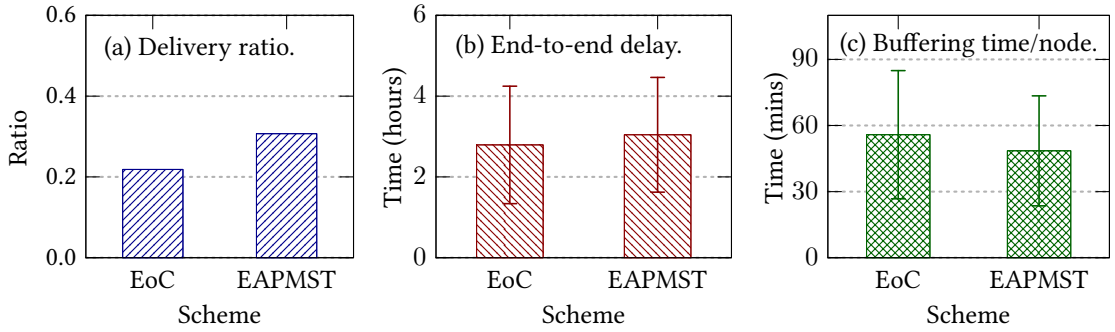
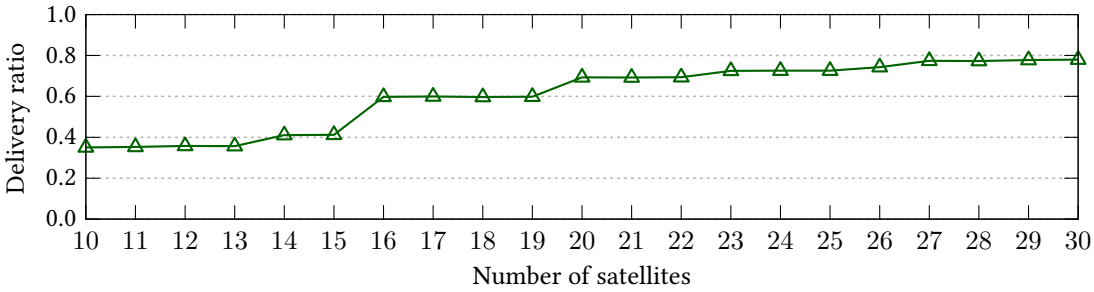
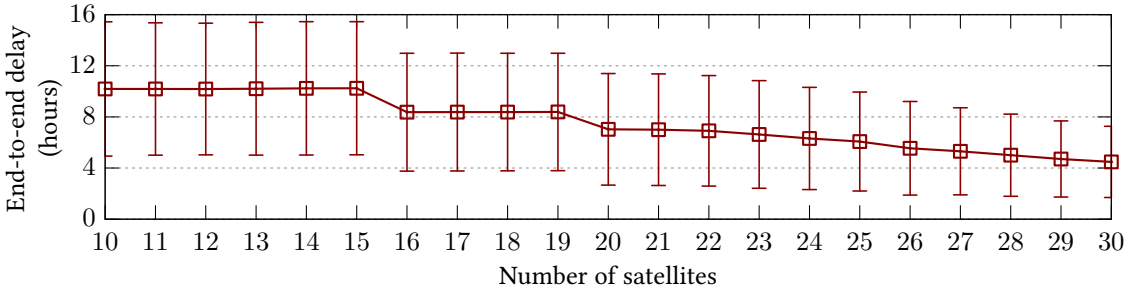


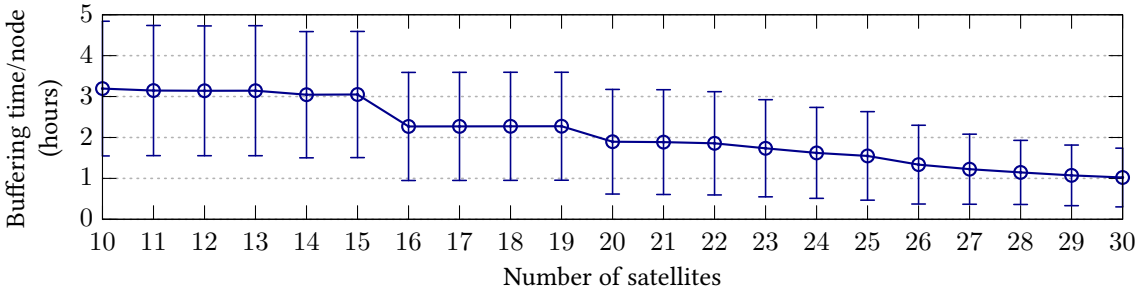
Figure 5.18: Performance of EAPMST on the DebriNet constellation.



(a) Delivery ratio vs. number of satellites.



(b) End-to-end delay vs. number of satellites.



(c) Per-node buffering time vs. number of satellites.

Figure 5.19: Performance of EAPMST on DebriNet with varying number of satellites.

nodes in space for a duration of 24 hours. It is clearly visible from Figure 5.19(a) that the delivery ratio increases with number of satellites, i.e., with each future satellite launch, an additional DebriNet node is deployed in space and subsequently the network can achieve a delivery ratio of 1 in future. A reduction in end-to-end delay can also be observed with increasing number of satellites in Figure 5.19(b). The decreasing pattern of per-node buffering time with increasing number of satellites in Figure 5.19(c) indicates the possibility of an end-to-end path between any two points at any time, i.e., zero per-node buffering time, in future.

The nature of satellite networks and their importance in next generation communication infrastructure necessitate a flexible network control with controlled buffering mechanism for satellites during link disruptions and node isolation. Our proposed SD-DTN framework offers a promising solution for managing traffic flows in SD-SNs depending on the network dynamics and user requirements. Also, the compatibility with existing SDN frameworks makes SD-DTN integration an easy task.

5.3 Other Application Domains of SD-DTNs

Apart from vehicular and satellite networks, SD-DTN is applicable to other classes of wireless networks which are prone to link disruptions. Two important domains include (*i*) tactical networks and (*ii*) emergency response networks.

5.3.1 Software Defined Tactical Networks

Networks designed for mission-critical operations constitute the class of tactical networks. The most common example of a tactical communication infrastructure includes the network deployed for military environments. Unlike other classes of wireless networks, tactical networks are characterized by specialized vehicles such as jeeps, tanks, helicopters, fighter jets, and Unmanned Aerial Vehicles (UAVs) along with ad hoc military tents at the battlefront. Furthermore, the vehicles are required to move on the uneven and uncertain terrains of the warfare environment. Besides the complexity of such an environment, the activities of a mission-critical operation are decided and executed at real-time depending on the actions from the opponents. Therefore, tactical communication infrastructures necessitate a flexible approach in network control, as offered by SDN, rather than the traditional deterministic algorithms [60, 175].

Figure 5.20 depicts a typical military infrastructure with a Command and Control Cen-

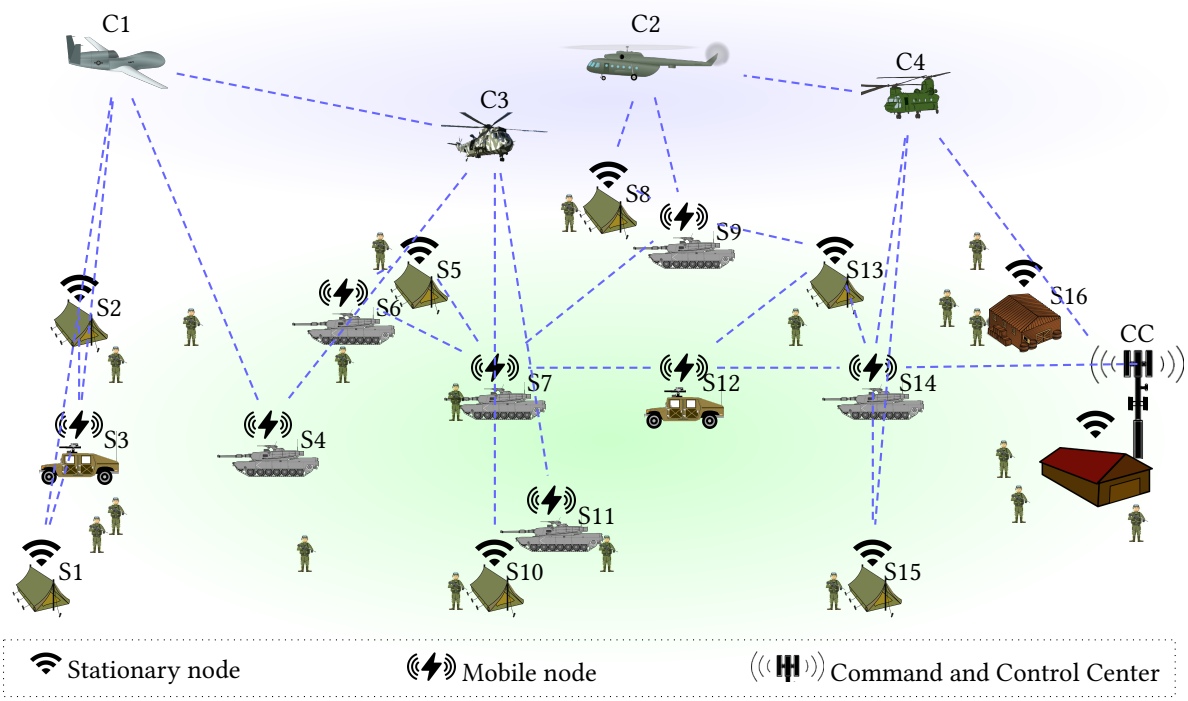


Figure 5.20: An example software defined tactical network scenario. The notations starting with the letter 'C' represent SDN controllers and those starting with the letter 'S' represent the switches.

ter (CC) as the focal point of the mission-critical decision making. The network includes ad hoc military tents and buildings as the only stationary nodes while jeeps, tanks, and aircraft constitute the mobile nodes. Each node is equipped with a wireless interface to communicate the mission-critical information to and from the CC. However, an always connected network becomes an infeasible option in military environments while ad hoc networking suits the need of the hour. From Figure 5.20, it is clear that the nodes form a wireless mesh network environment where the mobile nodes are crucial in establishing a communication path toward the CC. Moreover, the uncertainties in wireless medium along with the mobility of vehicles make the environment an SD-DTN scenario.

Due to the absence of an end-to-end path toward the CC, it is difficult for the controller deployed at the CC to send real-time flow-rules to the switches equipped on tents and vehicles. Therefore, airborne means such as drones, UAVs, helicopters, and aircraft equipped with local controllers are employed to provide flow-rules in real-time depending on the switches' location as shown in Figure 5.20. However, switches should be integrated with adequate mechanisms to connect to appropriate controllers so that the flow-rules can be retrieved and applied to the network flows at the earliest. In this context, our self-

configuration framework involving switch-initiated handoff can be a best-fit for switches and controllers to self-organize depending on the dynamic mission-critical requirements. Similar to control channel, data plane is also subject to link disruptions primarily due to the mobility of vehicles such as jeeps and tanks. In such cases, our proposed SD-DTN framework can help network engineers to store the packets at intermediate switches during the absence of end-to-end paths to/from the CC and forward the buffered packets when suitable links get established which can deliver the packets toward the intended destination with minimum delay.

5.3.2 Software Defined Emergency Response Networks

Networks designed with the mission of resolving issues emanating from a disaster are referred as emergency response networks or Incident Area Networks (IANs). Similar to tactical networks, emergency response networks are also designed in an ad hoc manner and are required to operate on real-time network control decisions for executing efficient emergency response services [18]. Examples of emergency response networks include NEARMesh [176] and UrgentMesh [177]. In addition, the network is exploited to gather information from the post-disaster environment, a crucial factor in minimizing the propagation of the disaster as we studied in [17] and [19].

Due to the possible absence or destruction of a communication infrastructure in a disaster affected region, especially in areas such as shanty towns, wireless nodes are deployed at selected locations to realize an ad hoc network through which the disaster information can be communicated to the command and control center. In addition to the fixed wireless communication points, mobile agents such as vehicles from safety and hospital services, and agents from non-voluntary organizations are employed to provide relief to the disaster affected people. Recent advancements in drone technology makes UAVs also an important component in the future emergency response systems [178–182]. The mobile agents involved in IANs are provided with sensors and wireless interfaces in order to gather and communicate the environmental parameters related to the aftermath of the disaster to the authorities for designing the subsequent disaster-relief operations. However, the mobility of vehicles and agents makes the ad hoc wireless environment prone to frequent link disruptions, thereby, leading to a DTN scenario.

Similar to tactical networks, the requirement of a flexible and real-time network control makes SDN an inevitable option for modern day emergency response systems [183, 184]. Therefore, our proposed SD-DTN framework along with the self-configuration

scheme suits software defined emergency response infrastructures for handling link disruptions in both SDN control channel and data plane. Due to the absence of an always end-to-end path between mobile agents and the CC, it is required to provide the flow-rules to the switches using airborne techniques, especially drones and UAVs, similar to tactical networks. Also, the inclusion of new agents as per the requirements of disaster relief operation scales the network, thereby, widening the scope of emergency response. However, the manual re-configuration of the new and existing switches and controllers on the addition or removal of nodes becomes a cumbersome task in environments with large number of nodes. In this context, our self-configuration scheme involving SD-OLSR protocol and switch-initiated handoff enables the network elements identify the SDN resources and self-configure depending on the real-time network control requirements. As far as the data plane is concerned, nodes in emergency response networks may require to forward the crucial and sensitive data such as patients' health information to the CC with high priority while the low sensitive packets need to be buffered during that time. Our SD-DTN framework can provide such a priority-based forwarding and buffering solution with the proposed STORE action. A UAV equipped with the controller can move near the switch and send real-time flow-rules involving a FORWARD action for the sensitive flows and a STORE action for the low priority flows, thereby, achieving controlled buffering/forwarding of different types of network flows.

Even though we discussed the four major wireless domains, SD-DTN is applicable to any class of wireless networks where link disruptions form a major hurdle in achieving the required QoS. SD-DTN's controlled buffering can even be used in SD-WLANs in enterprises to buffer packets at the access points to handle end-user mobility. Since the SD-DTN framework is compatible with the existing OpenFlow protocol, the STORE action can also be used in wired networks to buffer packets at the switches in order control network flows.

5.4 Observations and Discussion

The primary objective of the self-configuration scheme and the SD-DTN framework is to tolerate disruptions in software defined wireless networks, thereby, enabling the network to self-configure depending on the network dynamics and user requirements. Apart from the promising results from our experimental testbed demonstrating the efficacy of the proposed schemes, the simulation results from software defined vehicular and satellite environments show the applicability of our approach toward large-scale networking

environments involving different mobility patterns and varying link characteristics. In fact, SD-DTN and the proposed EAPMST are more suitable for satellite networks due to the predictable mobility pattern of satellites in their respective orbits. Such accurate predictions help in minimizing the buffering time of packets at the satellites by estimating the appropriate temporal paths. The DebriNet framework is important in this context and envisions a large-scale experimental SD-WN backbone in space employing the SD-DTN principles to buffer packets during link disruptions as well as during node isolation, major characteristics of satellite networks, and deliver the packets to the destination at the earliest. The results from DebriNet simulation using STK assert the improvement in performance provided by SD-DTN in terms of delivery ratio, end-to-end delay, and buffering time. Moreover, Figure 5.19 proves the ability of SD-DTN to adapt toward networks which are scalable in future.

Since SDN assumes the controller to possess the global network view to achieve fine-grained network control, we designed our self-configuration framework and SD-OLSR protocol with multiple physical controllers in order to maintain a consistent control plane during possible disruptions including link failure, controller failure, and network partitioning. In addition, the self-configuration schemes, SD-OLSR protocol, and the SD-DTN framework are independent of the number of switches and controllers within the network, thereby, making the framework suitable for large-scale networks as revealed from software defined satellite and vehicular networks.

5.5 Summary

In this chapter, we discussed the scope of our proposed self-configuration scheme and the SD-DTN framework in the emerging communication infrastructures. We started the discussion with our research contribution in designing a new metric *Effort* for placing roadside units in software defined vehicular networks considering road parameters such as road-length, road-type, and the importance of junctions. Using the metric *Effort*, an *Effort*-based Betweenness Centrality (EBC) scheme was developed for RSU placement. Simulation results showed the dominance of EBC scheme over the traditional road-length-based RSU placement scheme in improving the number of vehicle-RSU contacts while maintaining contact probability and contact time. Further, we shifted our focus toward analyzing the efficacy of EAPMST algorithm on software defined satellite networks considering two constellations: (*i*) *Constrained*-Iridium-NEXT and (*ii*) DebriNet. Controlled buffering with EAPMST provides higher delivery ratio and lower per-node buffering time

compared to the benchmark Earliest-on-Contact scheme. Apart from our contributions in SD-VNs and SD-SNs, we mentioned two major application domains for SD-DTN in Section 5.3: (*i*) software defined tactical networks and (*ii*) software defined emergency response networks. The experimental and simulation results on different classes of wireless networks demonstrated that our SD-DTN framework can be a promising candidate for resolving the issues emanating from link disruptions in wireless networks.

Chapter 6

Conclusions and Future Scope

> "*Disruption is a process, not an event, and innovations can only be disruptive relative to something else.*"
>
> — Clayton M. Christensen

> "*Without disruptions in life, where would we be?*"
>
> — Sarah Lynn Gadon

Wireless networks form inevitable components of the next generation communication infrastructures which are characterized by high data-rate, ultra-low latency, seamless connectivity, and all-time availability while handling heterogeneous networks involving enormous number of users, devices, and services along with their dynamic QoS requirements. Software defined networking is envisaged as the new pragmatic approach toward flexible network control in fine-tuning network flows depending on the dynamic user needs. However, the uncertainties in wireless medium as well as user mobility make the links in both SDN control and data channel disruption-prone, and fine-tuning network-flows in such environments becomes a strenuous task while achieving the aforementioned objectives. Therefore, we proposed a self-configuration scheme along with a software defined disruption tolerant networking framework to address the issues emanating from such disruption-prone wireless environments. The self-configuration scheme enables the switches and controllers in a software defined wireless network to self-adapt depending on the dynamic network conditions with the help of two controller handoff schemes, controller-initiated handoff and switch-initiated handoff, thereby, maintaining a consistent SDN control channel. We designed a software defined optimized link state routing protocol, the first of its kind in SD-WNs, for the automated SDN resource discovery and self-configuration. Further, we modelled the disruption-prone networks using temporal graphs and proposed an SD-DTN framework introducing the controlled buffering of

packets at the switches, using a new action STORE, for handling link disruptions in data plane. A new SD-DTN switch prototype and a buffering/routing control algorithm, called Earliest Arrival Path with Minimal Storage Time (EAPMST), was designed for demonstrating the proposed SD-DTN framework in achieving better network performance. The experimental results from disruption-prone SD-WMN testbeds revealed the efficacy of our self-configuration scheme and SD-DTN framework in sensing the network dynamics, self-configuring the network, and achieving the required fine-grained network control with controlled buffering. Analysis of our approach in the context of emerging application domains such as vehicular networks, satellite networks, and tactical and emergency response networks underlined the scope and advantage of our SD-DTN over the traditional networking approaches. Therefore, we believe that our SD-DTN framework can be a value addition for the next generation communication infrastructures in handling link disruptions, thereby, providing better quality-of-experience to the users.

6.1 Future Scope

Even though we verified the efficacy of our self-configuration scheme and SD-DTN framework on mobile wireless mesh environments, our approach is suitable for any class of wireless networks which are prone to link disruptions and, therefore, opening up a wide scope for advancements in the direction of next generation communication infrastructures. In fact, the compatibility with existing SDN architecture makes the integration of SD-DTN an easy task. From the perspective of the focus of this thesis, the following can be considered as the possible immediate future work:

1. Design of a hybrid handoff scheme for the self-configuration of SD-WNs by combining the controller and switch initiation techniques depending on the network state.

2. Design of SDN controller applications exploiting the advanced machine learning techniques such as recurrent neural networks for predicting the future states of the network, thereby, improving the performance of EAPMST algorithm.

3. Design of SD-DTN-controlled buffer management strategies considering the buffer space available at the nodes as well as the network flow characteristics.

4. Design of a light-weight SD-DTN framework for constrained SD-WNs such as wireless sensor networks and satellite networks.

Bibliography

[1] J. Walrand and P. Varaiya, "CHAPTER 7 - Wireless Networks," in *High-Performance Communication Networks*, 2nd ed., J. Walrand and P. Varaiya, Eds. Morgan Kaufmann, 2000, pp. 305–361. DOI: 10.1016/B978-0-08-050803-0.50012-5

[2] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5G wireless networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1617–1655, 3rd Quart. 2016. DOI: 10.1109/COMST.2016.2532458

[3] C. I, S. Han, Z. Xu, S. Wang, Q. Sun, and Y. Chen, "New paradigm of 5G wireless Internet," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 474–482, Mar. 2016. DOI: 10.1109/JSAC.2016.2525739

[4] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y. A. Zhang, "The roadmap to 6G: AI empowered wireless networks," *IEEE Communications Magazine*, vol. 57, no. 8, pp. 84–90, Aug. 2019. DOI: 10.1109/MCOM.2019.1900271

[5] Y. Zhao, J. Zhao, W. Zhai, S. Sun, D. Niyato, and K. Y. Lam, "A survey of 6G wireless communications: Emerging technologies," *arXiv preprint arXiv:2004.08549*, Jul. 2020. Online: https://arxiv.org/abs/2004.08549

[6] A. Goldsmith, *Wireless Communications*. Cambridge University Press, 2005. DOI: 10.1017/CBO9780511841224

[7] Open Networking Foundation, "Software-defined networking: The new norm for networks," *ONF White Paper*, vol. 2, pp. 2–6, Apr. 2012. Online: http://opennetworking.wpengine.com/wp-content/uploads/2011/09/wp-sdn-newnorm.pdf

[8] L. Yang, T. A. Anderson, R. Gopal, and R. Dantu, "Forwarding and Control Element Separation (ForCES) framework," RFC 3746, RFC Editor, Tech. Rep. 3746, Apr. 2004. DOI: 10.17487/RFC3746

[9] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. V. D. Merwe, "Design and implementation of a routing control platform," in *Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation (NSDI) - Volume 2*, May 2005, pp. 15–28. Online: http://dl.acm.org/citation.cfm?id=1251203.1251205

[10] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. V. D. Merwe, "The case for separating routing from routers," in *Proceedings of the ACM SIGCOMM Workshop on Future Directions in Network Architecture (FDNA)*, Aug. 2004, pp. 5–12. DOI: 10.1145/1016707.1016709

[11] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *SIGCOMM Computer Communications Review*, vol. 38, no. 2, pp. 69–74, Mar. 2008. DOI: 10.1145/1355734.1355746

[12] "OpenFlow switch specification version 1.0.0 (Wire protocol 0x01)," Open Networking Foundation, Tech. Rep., Dec. 2009. Online: https://opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.0.0.pdf

[13] "OpenFlow switch specification version 1.5.0 (Protocol version 0x06)," Open Networking Foundation, Tech. Rep., Dec. 2014. Online: https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.0.pdf

[14] K. Fall, "Retrospective on "a delay-tolerant network architecture for challenged internets"," *SIGCOMM Computer Communication Review*, vol. 49, no. 5, pp. 75–76, Nov. 2019. DOI: 10.1145/3371934.3371958

[15] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss, "Delay-tolerant networking architecture," Internet Requests for Comments, RFC Editor, RFC 4838, Apr. 2007. Online: http://www.rfc-editor.org/rfc/rfc4838.txt

[16] K. Scott and S. Burleigh, "Bundle protocol specification," Internet Requests for Comments, RFC Editor, RFC 5050, Nov. 2007. Online: http://www.rfc-editor.org/rfc/rfc5050.txt

[17] G. Jain, S. Babu, R. Raj, K. Benson, B. S. Manoj, and N. Venkatasubramanian, "On disaster information gathering in a complex shanty town terrain," in *Proceedings of*

the IEEE Global Humanitarian Technology Conference - South Asia Satellite (GHTC-SAS), Sep. 2014, pp. 147–153. DOI: 10.1109/GHTC-SAS.2014.6967574

[18] R. Raj, S. Babu, K. Benson, G. Jain, B. S. Manoj, and N. Venkatasubramanian, "Efficient path rescheduling of heterogeneous mobile data collectors for dynamic events in shanty town emergency response," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, Dec. 2015, pp. 1–7. DOI: 10.1109/GLOCOM.2015.7417610

[19] S. Babu, P. Rathod, and B. S. Manoj, "On optimizing information gathering in shanty town emergency response," in *Proceedings of the IEEE Region 10 Conference (TENCON)*, Oct. 2019, pp. 129–134. DOI: 10.1109/TENCON.2019.8929340

[20] A. V. Mamidi, S. Babu, and B. S. Manoj, "Dynamic multi-hop switch handoffs in software defined wireless mesh networks," in *Proceedings of the IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS)*, Dec. 2015, pp. 1–6. DOI: 10.1109/ANTS.2015.7413638

[21] A. Raniwala and T. Chiueh, "Evaluation of a wireless enterprise backbone network architecture," in *Proceedings of the 12th Annual IEEE Symposium on High Performance Interconnects*, Aug. 2004, pp. 98–104. DOI: 10.1109/CONECT.2004.1375211

[22] D. Bansal and S. Sofat, "Deployment and evaluation of IEEE 802.11 based wireless mesh networks in campus environment," in *Proceedings of the 4th ACM Workshop on Networked Systems for Developing Regions (NSDR)*, Jun. 2010, pp. 1–2. DOI: 10.1145/1836001.1836016

[23] M. Cesana, L. Fratta, M. Gerla, E. Giordano, and G. Pau, "C-VeT the UCLA campus vehicular testbed: Integration of VANET and mesh networks," in *Proceedings of the European Wireless Conference (EW)*, Apr. 2010, pp. 689–695. DOI: 10.1109/EW.2010.5483535

[24] L. A. D. Knob, R. P. Esteves, L. Z. Granville, and L. M. R. Tarouco, "Mitigating elephant flows in SDN-based IXP networks," in *Proceedings of the IEEE Symposium on Computers and Communications (ISCC)*, Jul. 2017, pp. 1352–1359. DOI: 10.1109/ISCC.2017.8024712

[25] Y. Huang, W. Shih, and J. Huang, "A classification-based elephant flow detection method using application round on SDN environments," in *Proceedings of the 19th*

Asia-Pacific Network Operations and Management Symposium (APNOMS), Sep. 2017, pp. 231–234. DOI: 10.1109/APNOMS.2017.8094140

[26] H. T. Zaw and A. H. Maw, "Elephant flow detection and delay-aware flow rerouting in software-defined network," in *Proceedings of the 9th International Conference on Information Technology and Electrical Engineering (ICITEE)*, Oct. 2017, pp. 1–6. DOI: 10.1109/ICITEED.2017.8250487

[27] X. Wang, A. Patil, and W. Wang, "VoIP over wireless mesh networks: Challenges and approaches," in *Proceedings of the 2nd Annual International Workshop on Wireless Internet (WICON)*, Aug. 2006, pp. 6–15. DOI: 10.1145/1234161.1234167

[28] A. Raniwala and T. Chiueh, "Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network," in *Proceedings of the 24th IEEE Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 3, Mar. 2005, pp. 2223–2234. DOI: 10.1109/INFCOM.2005.1498497

[29] A. Yarali, "Wireless mesh networking technology for commercial and industrial customers," in *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, May 2008, pp. 47–52. DOI: 10.1109/CCECE.2008.4564493

[30] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: An intellectual history of programmable networks," *SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, Apr. 2014. DOI: 10.1145/2602204.2602219

[31] N. A. Jagadeesan and B. Krishnamachari, "Software-defined networking paradigms in wireless networks: A survey," *ACM Computing Surveys*, vol. 47, no. 2, pp. 27:1–27:11, Nov. 2014. DOI: 10.1145/2655690

[32] I. T. Haque and N. A. Ghazaleh, "Wireless software defined networking: A survey and taxonomy," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2713–2737, 4th Quart. 2016. DOI: 10.1109/COMST.2016.2571118

[33] B. Dezfouli, V. Esmaeelzadeh, J. Sheth, and M. Radi, "A review of software-defined WLANs: Architectures and central control mechanisms," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 431–463, 1st Quart. 2019. DOI: 10.1109/COMST.2018.2868692

[34] P. H. Isolani, J. A. Wickboldt, C. B. Both, J. Rochol, and L. Z. Granville, "Interactive monitoring, visualization, and configuration of OpenFlow-based SDN," in *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM)*, May 2015, pp. 207–215. DOI: 10.1109/INM.2015.7140294

[35] Y. Zhang, L. Cui, W. Wang, and Y. Zhang, "A survey on software defined networking with multiple controllers," *Journal of Network and Computer Applications*, vol. 103, pp. 101–118, Feb. 2018. DOI: 10.1016/j.jnca.2017.11.015

[36] A. Sallahi and M. S. Hilaire, "Expansion model for the controller placement problem in software defined networks," *IEEE Communications Letters*, vol. 21, no. 2, pp. 274–277, Feb. 2017. DOI: 10.1109/LCOMM.2016.2621746

[37] B. P. R. Killi and S. V. Rao, "Optimal model for failure foresight capacitated controller placement in software-defined networks," *IEEE Communications Letters*, vol. 20, no. 6, pp. 1108–1111, Jun. 2016. DOI: 10.1109/LCOMM.2016.2550026

[38] B. P. R. Killi and S. V. Rao, "Capacitated next controller placement in software defined networks," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 514–527, Sep. 2017. DOI: 10.1109/TNSM.2017.2720699

[39] G. Wang, Y. Zhao, J. Huang, and Y. Wu, "An effective approach to controller placement in software defined wide area networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 344–355, Mar. 2018. DOI: 10.1109/TNSM.2017.2785660

[40] H. Huang, P. Li, S. Guo, and W. Zhuang, "Software-defined wireless mesh networks: Architecture and traffic orchestration," *IEEE Network*, vol. 29, no. 4, pp. 24–30, Jul. 2015. DOI: 10.1109/MNET.2015.7166187

[41] P. Dely, A. Kassler, and N. Bayer, "OpenFlow for wireless mesh networks," in *Proceedings of the 20th International Conference on Computer Communications and Networks (ICCCN)*, Jul. 2011, pp. 1–6. DOI: 10.1109/ICCCN.2011.6006100

[42] P. Patil, A. Hakiri, Y. Barve, and A. Gokhale, "Enabling software-defined networking for wireless mesh networks in smart environments," in *Proceedings of the 15th IEEE International Symposium on Network Computing and Applications (NCA)*, Oct. 2016, pp. 153–157. DOI: 10.1109/NCA.2016.7778610

[43] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, "Scalable flow-based networking with DIFANE," *SIGCOMM Computer Communication Review*, vol. 40, no. 4, pp. 351–362, Aug. 2010. DOI: 10.1145/1851275.1851224

[44] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: Scaling flow management for high-performance networks," *SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 254–265, Aug. 2011. DOI: 10.1145/2043164.2018466

[45] A. Detti, C. Pisa, S. Salsano, and N. B. Melazzi, "Wireless mesh software defined networks (wmSDN)," in *Proceedings of the 9th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct. 2013, pp. 89–95. DOI: 10.1109/WiMOB.2013.6673345

[46] T. Clausen, G. Hansen, L. Christensen, and G. Behrmann, "The optimized link state routing protocol, evaluation through experiments and simulation," in *Proceedings of the IEEE Symposium on Wireless Personal Mobile Communications*, Sep. 2001.

[47] T. N. Duc and E. Kamioka, "An extended SDN controller for handover in heterogeneous wireless network," in *Proceedings of the 21st Asia-Pacific Conference on Communications (APCC)*, Oct. 2015, pp. 332–337. DOI: 10.1109/APCC.2015.7412534

[48] S. H. Yeganeh and Y. Ganjali, "Turning the tortoise to the hare: An alternative perspective on event handling in SDN," in *Proceedings of the ACM International Workshop on Software-defined Ecosystems (BigSystem)*, Jun. 2014, pp. 29–32. DOI: 10.1145/2609441.2609642

[49] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. Kompella, "Towards an elastic distributed SDN controller," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, Aug. 2013, pp. 7–12. DOI: 10.1145/2491185.2491193

[50] D. Basu, A. A. Hussain, and S. F. Hasan, "A distributed mechanism for software-based mobility management," in *Proceedings of the 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, Aug. 2016, pp. 321–324. DOI: 10.1109/ICSESS.2016.7883076

[51] Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan, "Reliability-aware controller placement for software-defined networks," in *Proceedings of the*

IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), May 2013, pp. 672–675. Online: https://ieeexplore.ieee.org/document/6573050

[52] S. Guo, S. Yang, Q. Li, and Y. Jiang, "Towards controller placement for robust software-defined networks," in *Proceedings of the 34th IEEE International Performance Computing and Communications Conference (IPCCC)*, Dec. 2015, pp. 1–8. DOI: 10.1109/PCCC.2015.7410301

[53] M. T. I. U. Huque, G. Jourjon, and V. Gramoli, "Revisiting the controller placement problem," in *Proceedings of the 40th IEEE Conference on Local Computer Networks (LCN)*, Oct. 2015, pp. 450–453. DOI: 10.1109/LCN.2015.7366350

[54] M. T. I. U. Huque, W. Si, G. Jourjon, and V. Gramoli, "Large-scale dynamic controller placement," *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, pp. 63–76, Mar. 2017. DOI: 10.1109/TNSM.2017.2651107

[55] M. Labraoui, C. Chatzinakis, M. M. Boc, and A. Fladenmuller, "On addressing mobility issues in wireless mesh networks using software-defined networking," in *Proceedings of the 8th International Conference on Ubiquitous and Future Networks (ICUFN)*, Jul. 2016, pp. 903–908. DOI: 10.1109/ICUFN.2016.7536927

[56] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Technical Report CS-200006, Department of Computer Science, Duke University, Tech. Rep., 2000. Online: http://issg.cs.duke.edu/epidemic/epidemic.pdf

[57] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "MaxProp: Routing for vehicle-based disruption-tolerant networks," in *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM)*, Apr. 2006, pp. 1–11. DOI: 10.1109/INFOCOM.2006.228

[58] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 7, no. 3, pp. 19–20, Jul. 2003. DOI: 10.1145/961268.961272

[59] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and Wait: An efficient routing scheme for intermittently connected mobile networks," in *Proceedings of the ACM SIGCOMM Workshop on Delay-Tolerant Networking (WDTN)*, Aug. 2005, pp. 252–259. DOI: 10.1145/1080139.1080143

[60] I. Zacarias, L. P. Gaspary, A. Kohl, R. Q. A. Fernandes, J. M. Stocchero, and E. P. D. Freitas, "Combining software-defined and delay-tolerant approaches in last-mile tactical edge networking," *IEEE Communications Magazine*, vol. 55, no. 10, pp. 22–29, Oct. 2017. DOI: 10.1109/MCOM.2017.1700239

[61] T. Li, H. Zhou, H. Luo, and S. Yu, "SERvICE: A software defined framework for integrated space-terrestrial satellite communication," *IEEE Transactions on Mobile Computing*, vol. 17, no. 3, pp. 703–716, Mar. 2018. DOI: 10.1109/TMC.2017.2732343

[62] T. Li, H. Zhou, B. H. Feng, Q. Xu, and G. Li, "Software defined DTN-based satellite networks," Internet Engineering Task Force, Internet-Draft draft-li-dtn-sd-dtn-sat-net-05, Nov. 2018. Online: https://datatracker.ietf.org/doc/html/draft-li-dtn-sd-dtn-sat-net-05

[63] M. Fu and F. Wu, "Investigation of multipath routing algorithms in software defined networking," in *Proceedings of the International Conference on Green Informatics (ICGI)*, Aug. 2017, pp. 269–273. DOI: 10.1109/ICGI.2017.21

[64] T. D. Schepper, J. Struye, E. Zeljković, S. Latré, and J. Famaey, "Software-defined multipath-TCP for smart mobile devices," in *Proceedings of the 13th International Conference on Network and Service Management (CNSM)*, Nov. 2017, pp. 1–6. DOI: 10.23919/CNSM.2017.8256043

[65] K. Bao, J. D. Matyjas, F. Hu, and S. Kumar, "Intelligent software-defined mesh networks with link-failure adaptive traffic balancing," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 2, pp. 266–276, Jun. 2018. DOI: 10.1109/TCCN.2018.2790974

[66] G. Gupta, S. Babu, and B. S. Manoj, "Dual-mode TCP: An alternative approach for delay tolerant networks," in *Proceedings of the 23rd National Conference on Communications (NCC)*, Mar. 2017, pp. 1–6. DOI: 10.1109/NCC.2017.8077040

[67] A. Amokrane, R. Langar, R. Boutaba, and G. Pujolle, "Flow-based management for energy efficient campus networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 4, pp. 565–579, Dec. 2015. DOI: 10.1109/TNSM.2015.2501398

[68] E. Coronado, R. Riggio, J. Villalón, and A. Garrido, "Joint mobility management and multicast rate adaptation in software-defined enterprise WLANs," *IEEE Transactions*

on Network and Service Management, vol. 15, no. 2, pp. 625–637, Jun. 2018. DOI: 10.1109/TNSM.2018.2798296

[69] S. N. Hertiana, C. Hendrawan, and A. Kurniawan, "A joint approach to multipath routing and rate adaptation for congestion control in OpenFlow software defined network," in *Proceedings of the 1st International Conference on Wireless and Telematics (ICWT)*, Nov. 2015, pp. 1–6. DOI: 10.1109/ICWT.2015.7449209

[70] V. Nascimento, M. Moraes, R. Gomes, B. Pinheiro, A. Abelém, V. C. M. Borges, K. V. Cardoso, and E. Cerqueira, "Filling the gap between software defined networking and wireless mesh networks," in *Proceedings of the 10th International Conference on Network and Service Management (CNSM) and Workshop*, Nov. 2014, pp. 451–454. DOI: 10.1109/CNSM.2014.7014211

[71] I. Brito, S. Gramacho, I. Ferreira, M. Nazaré, L. Sampaio, and G. B. Figueiredo, "OpenWiMesh: A framework for software defined wireless mesh networks," in *Proceedings of the Brazilian Symposium on Computer Networks and Distributed Systems*, May 2014, pp. 199–206. DOI: 10.1109/SBRC.2014.24

[72] J. N. Martínez, J. Baranda, and J. M. Bafalluy, "A service-based model for the hybrid software defined wireless mesh backhaul of small cells," in *Proceedings of the 11th International Conference on Network and Service Management (CNSM)*, Nov. 2015, pp. 390–393. DOI: 10.1109/CNSM.2015.7367388

[73] S. Bera, S. Misra, and M. S. Obaidat, "Mobi-Flow: Mobility-aware adaptive flow-rule placement in software-defined access network," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1831–1842, Aug. 2019. DOI: 10.1109/TMC.2018.2868932

[74] D. Sajjadi, R. Ruby, M. Tanha, and J. Pan, "Fine-grained traffic engineering on SDN-aware Wi-Fi mesh networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 8, pp. 7593–7607, Aug. 2018. DOI: 10.1109/TVT.2018.2832010

[75] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5G: RAN, core network and caching solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3098–3130, 4th Quart. 2018. DOI: 10.1109/COMST.2018.2841349

[76] E. Coronado, S. N. Khan, and R. Riggio, "5G-EmPOWER: A software-defined networking platform for 5G radio access networks," *IEEE Transactions on Net-*

work and Service Management, vol. 16, no. 2, pp. 715–728, Jun. 2019. DOI: 10.1109/TNSM.2019.2908675

[77] A. Betzler, D. C. Mur, E. G. Villegas, I. Demirkol, and J. J. Aleixendri, "SODALITE: SDN wireless backhauling for dense 4G/5G small cell networks," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1709–1723, Dec. 2019. DOI: 10.1109/TNSM.2019.2930745

[78] W. Tan, J. Zhang, C. Peng, B. Xia, and Y. Kou, "SDN-enabled converged networks," *IEEE Wireless Communications*, vol. 21, no. 6, pp. 79–85, Dec. 2014. DOI: 10.1109/MWC.2014.7000975

[79] J. Bukhari and W. Yoon, "Multicasting in next-generation software-defined heterogeneous wireless networks," *IEEE Transactions on Broadcasting*, vol. 64, no. 4, pp. 915–921, Dec. 2018. DOI: 10.1109/TBC.2018.2834720

[80] D. Das, J. Bapat, and D. Das, "A dynamic QoS negotiation mechanism between wired and wireless SDN domains," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 1076–1085, Dec. 2017. DOI: 10.1109/TNSM.2017.2756819

[81] K. Fall, "A delay-tolerant network architecture for challenged Internets," in *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Aug. 2003, pp. 27–34. DOI: 10.1145/863955.863960

[82] S. Latifi, A. Durresi, and B. Cico, "Emulating enterprise network environments for fast transition to software-defined networking," in *Proceedings of the 3rd Mediterranean Conference on Embedded Computing (MECO)*, Jun. 2014, pp. 294–297. DOI: 10.1109/MECO.2014.6862721

[83] J. Chen, Y. Ma, H. Kuo, and W. Hung, "Enterprise visor: A software-defined enterprise network resource management engine," in *Proceedings of the IEEE/SICE International Symposium on System Integration*, Dec. 2014, pp. 381–384. DOI: 10.1109/SII.2014.7028068

[84] J. Chen, Y. Ma, H. Kuo, C. Yang, and W. Hung, "Software-defined network virtualization platform for enterprise network resource management," *IEEE Transactions on Emerging Topics in Computing*, vol. 4, no. 2, pp. 179–186, Apr. 2016. DOI: 10.1109/TETC.2015.2478757

[85] F. Kuliesius and V. Dangovas, "SDN enhanced campus network authentication and access control system," in *Proceedings of the 8th International Conference on Ubiquitous and Future Networks (ICUFN)*, Jul. 2016, pp. 894–899. DOI: 10.1109/ICUFN.2016.7536925

[86] C. Chen, Y. Lin, L. Yen, M. Chan, and C. Tseng, "Mobility management for low-latency handover in SDN-based enterprise networks," in *Proceedings of the IEEE Wireless Communications and Networking Conference*, Apr. 2016, pp. 1–6. DOI: 10.1109/WCNC.2016.7565105

[87] C. Lorenz, D. Hock, J. Scherer, R. Durner, W. Kellerer, S. Gebert, N. Gray, T. Zinner, and P. Tran-Gia, "An SDN/NFV-enabled enterprise network architecture offering fine-grained security policy enforcement," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 217–223, Mar. 2017. DOI: 10.1109/MCOM.2017.1600414CM

[88] Y. Cui, S. Xiao, C. Liao, I. Stojmenovic, and M. Li, "Data centers as software defined networks: Traffic redundancy elimination with wireless cards at routers," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 12, pp. 2658–2672, Dec. 2013. DOI: 10.1109/JSAC.2013.131207

[89] Y. Xu, Y. Yan, Z. Dai, and X. Wang, "A management model for SDN-based data center networks," in *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Apr. 2014, pp. 113–114. DOI: 10.1109/INFCOMW.2014.6849181

[90] S. Fang, Y. Yu, C. H. Foh, and K. M. M. Aung, "A loss-free multipathing solution for data center network using software-defined networking approach," *IEEE Transactions on Magnetics*, vol. 49, no. 6, pp. 2723–2730, Jun. 2013. DOI: 10.1109/TMAG.2013.2254703

[91] D. Li, Y. Shang, W. He, and C. Chen, "EXR: Greening data center network with software defined exclusive routing," *IEEE Transactions on Computers*, vol. 64, no. 9, pp. 2534–2544, Sep. 2015. DOI: 10.1109/TC.2014.2375233

[92] G. M. Saridis, S. Peng, Y. Yan, A. Aguado, B. Guo, M. Arslan, C. Jackson, W. Miao, N. Calabretta, F. Agraz, S. Spadaro, G. Bernini, N. Ciulli, G. Zervas, R. Nejabati, and D. Simeonidou, "Lightness: A function-virtualizable software defined data center network with all-optical circuit/packet switching," *Journal of Lightwave Technology*, vol. 34, no. 7, pp. 1618–1627, Apr. 2016. DOI: 10.1109/JLT.2015.2509476

[93] T. Zhu, D. Feng, F. Wang, Y. Hua, Q. Shi, J. Liu, Y. Cheng, and Y. Wan, "Efficient anonymous communication in SDN-based data center networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3767–3780, Dec. 2017. DOI: 10.1109/TNET.2017.2751616

[94] Y. Wang and S. You, "An efficient route management framework for load balance and overhead reduction in SDN-based data center networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1422–1434, Dec. 2018. DOI: 10.1109/TNSM.2018.2872054

[95] J. M. Wang, Y. Wang, X. Dai, and B. Bensaou, "SDN-based multi-class QoS guarantee in inter-data center communications," *IEEE Transactions on Cloud Computing*, vol. 7, no. 1, pp. 116–128, Jan. 2019. DOI: 10.1109/TCC.2015.2491930

[96] C. Chuang, Y. Yu, and A. Pang, "Flow-aware routing and forwarding for SDN scalability in wireless data centers," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1676–1691, Dec. 2018. DOI: 10.1109/TNSM.2018.2865166

[97] X. Xu, H. Zhang, X. Dai, Y. Hou, X. Tao, and P. Zhang, "SDN based next generation mobile network with service slicing and trials," *China Communications*, vol. 11, no. 2, pp. 65–77, Feb. 2014. DOI: 10.1109/CC.2014.6821738

[98] P. K. Agyapong, M. Iwamura, D. Staehle, W. Kiess, and A. Benjebbour, "Design considerations for a 5G network architecture," *IEEE Communications Magazine*, vol. 52, no. 11, pp. 65–75, Nov. 2014. DOI: 10.1109/MCOM.2014.6957145

[99] H. Li, M. Dong, and K. Ota, "Control plane optimization in software-defined vehicular ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 10, pp. 7895–7904, Oct. 2016. DOI: 10.1109/TVT.2016.2563164

[100] K. Wang, Y. Wang, D. Zeng, and S. Guo, "An SDN-based architecture for next-generation wireless networks," *IEEE Wireless Communications*, vol. 24, no. 1, pp. 25–31, Feb. 2017. DOI: 10.1109/MWC.2017.1600187WC

[101] F. Z. Yousaf, M. Bredel, S. Schaller, and F. Schneider, "NFV and SDN—key technology enablers for 5G networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2468–2478, Nov. 2017. DOI: 10.1109/JSAC.2017.2760418

[102] D. A. Temesgene, J. N. Martinez, and P. Dini, "Softwarization and optimization for sustainable future mobile networks: A survey," *IEEE Access*, vol. 5, pp. 25 421–25 436, Nov. 2017. DOI: 10.1109/ACCESS.2017.2771938

[103] S. Xu, X. Wang, and M. Huang, "Software-defined next-generation satellite networks: Architecture, challenges, and solutions," *IEEE Access*, vol. 6, pp. 4027–4041, Jan. 2018. DOI: 10.1109/ACCESS.2018.2793237

[104] B. Deng, C. Jiang, H. Yao, S. Guo, and S. Zhao, "The next generation heterogeneous satellite communication networks: Integration of resource management and deep reinforcement learning," *IEEE Wireless Communications*, vol. 27, no. 2, pp. 105–111, Apr. 2020. DOI: 10.1109/MWC.001.1900178

[105] Z. Lv and W. Xiu, "Interaction of edge-cloud computing based on SDN and NFV for next generation IoT," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5706–5712, Jul. 2020. DOI: 10.1109/JIOT.2019.2942719

[106] O. S. Oubbati, M. Atiquzzaman, T. A. Ahanger, and A. Ibrahim, "Softwarization of UAV networks: A survey of applications and future trends," *IEEE Access*, vol. 8, pp. 98 073–98 125, May 2020. DOI: 10.1109/ACCESS.2020.2994494

[107] W. Zang, Z. Jin, and J. Lan, "An SDN based fast rerouting mechanism for elephant flows in DCN," in *Proceedings of the 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, Nov. 2017, pp. 363–366. DOI: 10.1109/ICSESS.2017.8342933

[108] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking (MobiCom)*, Sep. 2003, pp. 134–146. DOI: 10.1145/938985.939000

[109] J. L. Sobrinho, "Network routing with path vector protocols: Theory and applications," in *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Aug. 2003, pp. 49–60. DOI: 10.1145/863955.863963

[110] T. Clausen and P. Jacquet, "Optimized link state routing protocol (OLSR)," Internet Requests for Comments, RFC Editor, RFC 3626, Oct. 2003. Online: http://www.rfc-editor.org/rfc/rfc3626.txt

[111] "OpenFlow switch specification version 1.3.0 (Wire protocol 0x04)," Open Networking Foundation, Tech. Rep., Jul. 2012. Online: https://opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf

[112] RYU Project Team, "RYU SDN framework," Tech. Rep., 2015. Online: https://osrg.github.io/ryu-book/en/Ryubook.pdf

[113] "ping(8) - Linux man page," https://linux.die.net/man/8/ping, (Accessed on 01/01/2020).

[114] J. A. Wickboldt, W. P. D. Jesus, P. H. Isolani, C. B. Both, J. Rochol, and L. Z. Granville, "Software-defined networking: Management requirements and challenges," *IEEE Communications Magazine*, vol. 53, no. 1, pp. 278–285, Jan. 2015. DOI: 10.1109/MCOM.2015.7010546

[115] S. Babu, A. Rajeev, and B. S. Manoj, "A medium-term disruption tolerant SDN for wireless TCP/IP networks," *IEEE Transactions on Network and Service Management*, pp. 1–17, Sep. 2020. DOI: 10.1109/TNSM.2020.3023889

[116] P. Holme and J. Saramäki, "Temporal networks," *Physics Reports*, vol. 519, no. 3, pp. 97–125, Oct. 2012. DOI: 10.1016/j.physrep.2012.03.001

[117] P. Basu, A. B. Noy, R. Ramanathan, and M. P. Johnson, "Modeling and analysis of time-varying graphs," *CoRR*, vol. abs/1012.0260, Dec. 2010. Online: http://arxiv.org/abs/1012.0260

[118] F. Kaup, F. Michelinakis, N. Bui, J. Widmer, K. Wac, and D. Hausheer, "Assessing the implications of cellular network performance on mobile content access," *IEEE Transactions on Network and Service Management*, vol. 13, no. 2, pp. 168–180, Jun. 2016. DOI: 10.1109/TNSM.2016.2544402

[119] P. Kong and A. Sluzek, "Average packet delay analysis for a mobile user in a two-tier heterogeneous cellular network," *IEEE Systems Journal*, vol. 11, no. 4, pp. 2726–2736, Dec. 2017. DOI: 10.1109/JSYST.2015.2438096

[120] H. Farooq and A. Imran, "Spatiotemporal mobility prediction in proactive self-organizing cellular networks," *IEEE Communications Letters*, vol. 21, no. 2, pp. 370–373, Feb. 2017. DOI: 10.1109/LCOMM.2016.2623276

[121] H. Tabassum, M. Salehi, and E. Hossain, "Fundamentals of mobility-aware performance characterization of cellular networks: A tutorial," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2288–2308, 3rd Quart. 2019. DOI: 10.1109/COMST.2019.2907195

[122] F. Daneshgaran, M. Laddomada, F. Mesiti, M. Mondin, and M. Zanolo, "Saturation throughput analysis of IEEE 802.11 in the presence of non ideal transmission channel and capture effects," *IEEE Transactions on Communications*, vol. 56, no. 7, pp. 1178–1188, Jul. 2008. DOI: 10.1109/TCOMM.2008.060397

[123] D. Gong and Y. Yang, "On-line AP association algorithms for 802.11n WLANs with heterogeneous clients," *IEEE Transactions on Computers*, vol. 63, no. 11, pp. 2772–2786, Nov. 2014. DOI: 10.1109/TC.2013.156

[124] P. Rahimzadeh and F. Ashtiani, "Analytical evaluation of saturation throughput of a cognitive WLAN overlaid on a time-scheduled OFDMA network," *IEEE Transactions on Mobile Computing*, vol. 16, no. 3, pp. 634–647, Mar. 2017. DOI: 10.1109/TMC.2016.2567387

[125] Y. Wu, S. M. Das, and R. Chandra, "Routing with a Markovian metric to promote local mixing," in *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM)*, May 2007, pp. 2381–2385. DOI: 10.1109/INFCOM.2007.285

[126] N. Weragama, J. Jun, J. Mitro, and D. P. Agrawal, "Modeling and performance of a mesh network with dynamically appearing and disappearing femtocells as additional Internet gateways," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 5, pp. 1278–1288, May 2014. DOI: 10.1109/TPDS.2013.129

[127] L. Lei, J. Zhou, X. Chen, L. Qi, and S. Cai, "Modelling and analysing medium access delay for differentiated services in IEEE 802.11s wireless mesh networks," *IET Networks*, vol. 1, no. 2, pp. 91–99, Jun. 2012. DOI: 10.1049/iet-net.2012.0010

[128] Y. Xie, C. Wang, Y. Wang, and Y. Zheng, "An efficient pub/sub protocol for WMNs based on heuristic Markov chain," in *Proceedings of the 9th IEEE International Conference on Networking, Sensing and Control*, Apr. 2012, pp. 203–208. DOI: 10.1109/ICNSC.2012.6204917

[129] X. Zhao, J. Guo, C. T. Chou, A. Misra, and S. Jha, "A high-throughput routing metric for reliable multicast in multi-rate wireless mesh networks," in *Proceedings of the IEEE INFOCOM*, Apr. 2011, pp. 2042–2050. DOI: 10.1109/INFCOM.2011.5935012

[130] Y. Zhu, Y. Wu, and B. Li, "Trajectory improves data delivery in urban vehicular networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 4, pp. 1089–1100, Apr. 2014. DOI: 10.1109/TPDS.2013.118

[131] S. Bharati and W. Zhuang, "CRB: Cooperative relay broadcasting for safety applications in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 12, pp. 9542–9553, Dec. 2016. DOI: 10.1109/TVT.2016.2598488

[132] G. Dubosarskii, S. L. Primak, and X. Wang, "Evolution of vehicle network on a highway," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 9088–9097, Sep. 2019. DOI: 10.1109/TVT.2019.2927389

[133] M. Z. Alam, F. S. Abkenar, I. Adhicandra, S. Murali, and A. Jamalipour, "Low-delay path selection for cluster-based buffer-aided vehicular communications," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 9356–9363, Sep. 2020. DOI: 10.1109/TVT.2020.2976926

[134] S. F. Hasan, X. Ding, N. H. Siddique, and S. Chakraborty, "Measuring disruption in vehicular communications," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 1, pp. 148–159, Jan. 2011. DOI: 10.1109/TVT.2010.2087780

[135] H. Wu, J. Cheng, S. Huang, Y. Ke, Y. Lu, and Y. Xu, "Path problems in temporal graphs," *Proceedings of the VLDB Endowment*, vol. 7, no. 9, pp. 721–732, May 2014. DOI: 10.14778/2732939.2732945

[136] B. B. Xuan, A. Ferreira, and A. Jarry, "Computing shortest, fastest, and foremost journeys in dynamic networks," *International Journal of Foundations of Computer Science*, vol. 14, no. 02, pp. 267–285, Apr. 2003. DOI: 10.1142/S0129054103001728

[137] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social Networks*, vol. 1, no. 3, pp. 215–239, Jan. 1978. DOI: 10.1016/0378-8733(78)90021-7

[138] A. Pentland, R. Fletcher, and A. Hasson, "DakNet: Rethinking connectivity in developing nations," *Computer*, vol. 37, no. 1, pp. 78–83, Jan. 2004. DOI: 10.1109/MC.2004.1260729

[139] X. Ge, Z. Li, and S. Li, "5G software defined vehicular networks," *IEEE Communications Magazine*, vol. 55, no. 7, pp. 87–93, Jul. 2017. DOI: 10.1109/MCOM.2017.1601144

[140] C. Chen, T. H. Luan, X. Guan, N. Lu, and Y. Liu, "Connected vehicular transportation: Data analytics and traffic-dependent networking," *IEEE Vehicular Technology Magazine*, vol. 12, no. 3, pp. 42–54, Sep. 2017. DOI: 10.1109/MVT.2016.2645318

[141] R. Almeida, R. Oliveira, M. Luis, C. Senna, and S. Sargento, "Forwarding strategies for future mobile smart city networks," in *Proceedings of the 87th IEEE Vehicular Technology Conference (VTC Spring)*, Jun. 2018, pp. 1–7. DOI: 10.1109/VTCSpring.2018.8417757

[142] C. E. Palazzi, F. Pezzoni, and P. M. Ruiz, "Delay-bounded data gathering in urban vehicular sensor networks," *Pervasive and Mobile Computing (Special Issue on Wide-Scale Vehicular Sensor Networks and Mobile Sensing)*, vol. 8, no. 2, pp. 180–193, Apr. 2012. DOI: 10.1016/j.pmcj.2011.06.008

[143] J. C. Castillo, S. Zeadally, and J. A. G. Ibañez, "Internet of Vehicles: Architecture, protocols, and security," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3701–3709, Oct. 2018. DOI: 10.1109/JIOT.2017.2690902

[144] M. Gerla, E. Lee, G. Pau, and U. Lee, "Internet of Vehicles: From intelligent grid to autonomous cars and vehicular clouds," in *Proceedings of the IEEE World Forum on Internet of Things (WF-IoT)*, Mar. 2014, pp. 241–246. DOI: 10.1109/WF-IoT.2014.6803166

[145] I. Ku, Y. Lu, M. Gerla, R. L. Gomes, F. Ongaro, and E. Cerqueira, "Towards software-defined VANET: Architecture and services," in *Proceedings of the 13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*, Jun. 2014, pp. 103–110. DOI: 10.1109/MedHocNet.2014.6849111

[146] N. B. Truong, G. M. Lee, and Y. G. Doudane, "Software defined networking-based vehicular adhoc network with fog computing," in *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM)*, May 2015, pp. 1202–1207. DOI: 10.1109/INM.2015.7140467

[147] OpenStreetMap, "Key:highway - OpenStreetMap Wiki," https://wiki.openstreetmap.org/wiki/Key:highway, (Accessed on 05/30/2019).

[148] S. Babu and B. S. Manoj, "Toward a type-based analysis of road networks," *ACM Transactions on Spatial Algorithms and Systems*, vol. 6, no. 4, pp. 28:1–28:45, Aug. 2020. DOI: 10.1145/3397579

[149] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, no. 3&4, pp. 128–138, Dec. 2012. Online: http://elib.dlr.de/80483/

[150] "DUAROUTER - SUMO documentation," https://sumo.dlr.de/docs/DUAROUTER.html, (Accessed on 11/04/2019).

[151] A. Kchiche and F. Kamoun, "Centrality-based access-points deployment for vehicular networks," in *Proceedings of the 17th International Conference on Telecommunications*, Apr. 2010, pp. 700–706. DOI: 10.1109/ICTEL.2010.5478800

[152] OpenStreetMap, "Key:maxspeed - OpenStreetMap Wiki," https://wiki.openstreetmap.org/wiki/Key:maxspeed, (Accessed on 09/01/2020).

[153] S. Babu, G. Jain, and B. S. Manoj, "Urban Delay Tolerant Network Simulator (UDTNSim v0.1)," *CoRR*, vol. abs/1709.05645, Sep. 2017. Online: http://arxiv.org/abs/1709.05645

[154] M. Rogers, "Global reach - Iridium leverages worldwide coverage," *Energy Northern Perspective*, Sep. 2019, (Accessed on 01/21/2020). Online: https://energynorthern.com/2019/09/13/global-reach-iridium-leverages-worldwide-coverage

[155] J. Ray, "Globalstar constellation completed with Delta launch," https://spaceflightnow.com/delta/d276/000208launch.html, Feb. 2000, (Accessed on 01/21/2020).

[156] "Starlink," https://www.starlink.com/, 2019, (Accessed on 01/21/2020).

[157] "OneWeb satellites constellation: Connection for people all over the globe - Telecommunications - Airbus," https://www.airbus.com/space/telecommunications-satellites/oneweb-satellites-connection-for-people-all-over-the-globe.html, 2020, (Accessed on 01/21/2020).

[158] K. Maine, C. Devieux, and P. Swan, "Overview of IRIDIUM satellite network," in *Proceedings of WESCON'95*, Nov. 1995, pp. 483–490. DOI: 10.1109/WESCON.1995.485428

[159] R. A. Wiedeman and A. J. Viterbi, "The Globalstar mobile satellite system for worldwide personal communications," in *Proceedings of the 3rd International Mobile Satellite Conference (IMSC 1993)*, Jan. 1993, pp. 291–296. Online: https://ntrs.nasa.gov/api/citations/19940018313/downloads/19940018313.pdf

[160] "Telesat LEO," https://www.telesat.com/leo-satellites/, (Accessed on 12/17/2020).

[161] I. D. Portillo, B. G. Cameron, and E. F. Crawley, "A technical comparison of three low earth orbit satellite constellation systems to provide global broadband," *Acta Astronautica*, vol. 159, pp. 123–135, Jun. 2019. DOI: https://doi.org/10.1016/j.actaastro.2019.03.040

[162] S. K. Dao, "The role of satellite networks in the 21st century," in *Internetworking and Computing Over Satellite Networks*, Y. Zhang, Ed. Springer US, 2003, pp. 1–12. DOI: 10.1007/978-1-4615-0431-3_1

[163] *Telstra Satellite Services: Bringing the world to your doorstep*, https://www.telstraglobal.com/images/assets/products/resources/SatelliteDataServices-Datasheet.pdf, Telstra Global, Mar. 2017, (Accessed on 01/21/2020).

[164] J. Bao, B. Zhao, W. Yu, Z. Feng, C. Wu, and Z. Gong, "OpenSAN: A software-defined satellite network architecture," in *Proceedings of the ACM Conference on SIGCOMM*, Aug. 2014, pp. 347–348. DOI: 10.1145/2619239.2631454

[165] R. Suraj, S. Babu, D. Dalai, and B. S. Manoj, "DebriNet: An opportunistic software defined networking framework over PSLV debris," in *Proceedings of the IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, Dec. 2019, pp. 1–6. DOI: 10.1109/ANTS47819.2019.9118082

[166] L. Wood, "Satellite constellation networks," in *Internetworking and Computing Over Satellite Networks*, Y. Zhang, Ed. Boston, MA: Springer US, 2003, pp. 13–34. DOI: 10.1007/978-1-4615-0431-3_2

[167] "Systems Took Kit (STK)," (Accessed on 07/21/2019). Online: http://www.agi.com/products

[168] "Indian Space Research Organisation - Government of India," https://www.isro.gov.in/, (Accessed on 12/18/2020).

[169] "Iridium NEXT - Satellite Missions - eoPortal Directory," https://directory.eoportal.org/web/eoportal/satellite-missions/i/iridium-next, (Accessed on 11/06/2020).

[170] J. C. Liou and N. L. Johnson, "Risks in space from orbiting debris," *Science*, vol. 311, no. 5759, pp. 340–341, Jan. 2006. DOI: 10.1126/science.1121337

[171] H. Klinkrad, "Space debris," in *Encyclopedia of Aerospace Engineering*. American Cancer Society, 2010. DOI: https://doi.org/10.1002/9780470686652.eae325

[172] "PSLV - ISRO," https://www.isro.gov.in/launchers/pslv, (Accessed on 11/18/2020).

[173] "TeLEOS-1 - Satellite Missions - eoPortal Directory," https://directory.eoportal.org/web/eoportal/satellite-missions/t/teleos-1, (Accessed on 11/18/2020).

[174] "LoRaWAN®distance world record broken, twice. $766\,\mathrm{km}$ (476 miles) using $25\,\mathrm{mW}$ transmission power," https://www.thethingsnetwork.org/article/lorawan-distance-world-record#:~:text=The%20new%20world%20record%20approaches,in%20the%20868Mhz%20ISM%20band., (Accessed on 11/10/2020).

[175] K. Chen, N. Lv, S. Zhao, X. Wang, and J. Zhao, "A scheme for improving the communications efficiency between the control plane and data plane of the SDN-enabled airborne tactical network," *IEEE Access*, vol. 6, pp. 37 286–37 301, Jul. 2018. DOI: 10.1109/ACCESS.2018.2852707

[176] T. A. Hadhrami and F. Saeed, "NEARMesh: Network environment aware routing in a wireless mesh network for emergency-response," in *Proceedings of the International Conference on Research and Innovation in Information Systems (ICRIIS)*, Jul. 2017, pp. 1–6. DOI: 10.1109/ICRIIS.2017.8002448

[177] T. A. Hadhrami, Q. Wang, M. Crowe, and C. Grecos, "UrgentMesh: Wireless mesh networks with DVB-satellite for emergency management," in *Proceedings of the 3rd International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, Oct. 2011, pp. 1–6. Online: https://ieeexplore.ieee.org/abstract/document/6078963

[178] W. Jin, J. Yang, Y. Fang, and W. Feng, "Research on application and deployment of UAV in emergency response," in *Proceedings of the 10th IEEE International Conference on Electronics Information and Emergency Communication (ICEIEC)*, Jul. 2020, pp. 277–280. DOI: 10.1109/ICEIEC49280.2020.9152338

[179] A. Anastasiou, P. Kolios, C. Panayiotou, and K. Papadaki, "Swarm path planning for the deployment of drones in emergency response missions," in *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS)*, Sep. 2020, pp. 456–465. DOI: 10.1109/ICUAS48674.2020.9213876

[180] Z. Huang, C. Chen, and M. Pan, "Multiobjective UAV path planning for emergency information collection and transmission," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6993–7009, Aug. 2020. DOI: 10.1109/JIOT.2020.2979521

[181] H. Shakhatreh, A. Khreishah, and B. Ji, "UAVs to the rescue: Prolonging the lifetime of wireless devices under disaster situations," *IEEE Transactions on Green Communications and Networking*, vol. 3, no. 4, pp. 942–954, Dec. 2019. DOI: 10.1109/TGCN.2019.2930642

[182] A. Ranjan, B. Panigrahi, H. K. Rath, P. Misra, and A. Simha, "LTE-CAS: LTE-based criticality aware scheduling for UAV assisted emergency response," in *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WORKSHOPS)*, Apr. 2018, pp. 894–899. DOI: 10.1109/INFCOMW.2018.8406949

[183] M. Rahouti, K. Xiong, T. Chin, and P. Hu, "SDN-ERS: A timely software defined networking framework for emergency response systems," in *Proceedings of the IEEE International Science of Smart City Operations and Platforms Engineering in Partnership with Global City Teams Challenge (SCOPE-GCTC)*, Apr. 2018, pp. 18–23. DOI: 10.1109/SCOPE-GCTC.2018.00010

[184] M. Rahouti, K. Xiong, T. Chin, P. Hu, and D. D. Oliveira, "A preemption-based timely software defined networking framework for emergency response traffic delivery," in *Proceedings of the 21st IEEE International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, Aug. 2019, pp. 452–459. DOI: 10.1109/HPCC/SmartCity/DSS.2019.00074

List of Publications

Refereed Journals

1. **Sarath Babu**, A. Rajeev, and B. S. Manoj, "A medium-term disruption tolerant SDN for wireless TCP/IP networks," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2318–2334, Dec. 2020. DOI: 10.1109/TNSM.2020.3023889

2. **Sarath Babu** and B. S. Manoj, "Toward a type-based analysis of road networks," *ACM Transactions on Spatial Algorithms and Systems*, vol. 6, no. 4, pp. 28:1–28:45, Aug. 2020. DOI: 10.1145/3397579

3. **Sarath Babu**, P. V. Mithun, and B. S. Manoj, "A novel framework for resource discovery and self-configuration in software defined wireless mesh networks," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 132–146, Mar. 2020. DOI: 10.1109/TNSM.2019.2922107

Refereed Conferences

1. **Sarath Babu**, I. Ghosh, and B. S. Manoj, "Effort: A new metric for roadside unit placement in 5G enabled vehicular networks," in *Proceedings of the 3rd IEEE 5G World Forum (5GWF)*, Sep. 2020, pp. 263–268. DOI: 10.1109/5GWF49715.2020.9221228

2. **Sarath Babu**, P. Rathod, and B. S. Manoj, "On optimizing information gathering in shanty town emergency response," in *Proceedings of the IEEE Region 10 Conference (TENCON)*, Oct. 2019, pp. 129–134. DOI: 10.1109/TENCON.2019.8929340

3. **Sarath Babu** and B. S. Manoj, "On the topology of Indian and Western road networks," in *Proceedings of the 8th International Conference on Communication Systems and Networks (COMSNETS)*, Jan. 2016, pp. 1–6. DOI: 10.1109/COMSNETS.2016.7440027

Appendix A

Information on Satellite Constellations

Table A.1: Satellites used in the *Constrained*-Iridium-NEXT constellation.

Sl.No.	Name	Perigee (km)	Apogee (km)	Inclination (degrees)	Period (mins)	Semi major axis (km)	Launch Date (YYYY-MM-DD)	NORAD ID	CASPAR ID
1	IRIDIUM 100	783.3	786.3	86.4	100.4	7,155	2017-10-09	42956	2017-061B
2	IRIDIUM 102	783.6	786.1	86.4	100.4	7,155	2017-01-14	41920	2017-003D
3	IRIDIUM 103	783.4	786.2	86.4	100.4	7,155	2017-01-14	41918	2017-003B
4	IRIDIUM 104	783.2	786.5	86.4	100.4	7,155	2017-01-14	41922	2017-003F
5	IRIDIUM 105	782.9	786.7	86.4	100.4	7,155	2017-01-14	41921	2017-003E
6	IRIDIUM 106	783.1	786.5	86.4	100.4	7,155	2017-01-14	41917	2017-003A
7	IRIDIUM 107	783.3	786.3	86.4	100.4	7,155	2017-10-09	42960	2017-061F
8	IRIDIUM 108	783.1	786.5	86.4	100.4	7,155	2017-01-14	41924	2017-003H
9	IRIDIUM 109	783.4	786.2	86.4	100.4	7,155	2017-01-14	41919	2017-003C
10	IRIDIUM 110	783.2	786.4	86.4	100.4	7,155	2018-05-22	43481	2018-047F
11	IRIDIUM 111	783.4	786.2	86.4	100.4	7,155	2017-01-14	41926	2017-003K
12	IRIDIUM 112	783.5	786.1	86.4	100.4	7,155	2017-01-14	41925	2017-003J
13	IRIDIUM 113	783.3	786.4	86.4	100.4	7,155	2017-06-25	42803	2017-039A
14	IRIDIUM 114	783.3	786.3	86.4	100.4	7,155	2017-01-14	41923	2017-003G
15	IRIDIUM 115	754.1	755.6	86.5	99.8	7,125	2017-06-25	42806	2017-039D
16	IRIDIUM 116	783.5	786.1	86.4	100.4	7,155	2017-12-23	43072	2017-083C
17	IRIDIUM 117	783.1	786.6	86.4	100.4	7,155	2017-06-25	42808	2017-039F
18	IRIDIUM 118	783.6	786.0	86.4	100.4	7,155	2017-06-25	42807	2017-039E
19	IRIDIUM 119	783.5	786.1	86.4	100.4	7,155	2017-10-09	42959	2017-061E
20	IRIDIUM 120	783.2	786.4	86.4	100.4	7,155	2017-06-25	42805	2017-039C
21	IRIDIUM 121	783.4	786.2	86.4	100.4	7,155	2017-06-25	42812	2017-039K
22	IRIDIUM 122	783.3	786.3	86.4	100.4	7,155	2017-10-09	42957	2017-061C
23	IRIDIUM 123	783.5	786.1	86.4	100.4	7,155	2017-06-25	42804	2017-039B
24	IRIDIUM 124	753.0	756.6	86.5	99.8	7,125	2017-06-25	42810	2017-039H
25	IRIDIUM 125	783.3	786.3	86.4	100.4	7,155	2017-10-09	42964	2017-061K
26	IRIDIUM 126	783.4	786.3	86.4	100.4	7,155	2017-06-25	42809	2017-039G
27	IRIDIUM 128	783.4	786.2	86.4	100.4	7,155	2017-06-25	42811	2017-039J
28	IRIDIUM 129	783.4	786.2	86.4	100.4	7,155	2017-10-09	42958	2017-061D
29	IRIDIUM 130	783.4	786.2	86.4	100.4	7,155	2017-12-23	43073	2017-083D
30	IRIDIUM 131	783.3	786.3	86.4	100.4	7,155	2017-12-23	43079	2017-083K
31	IRIDIUM 132	783.2	786.5	86.4	100.4	7,155	2017-10-09	42961	2017-061G
32	IRIDIUM 133	783.4	786.2	86.4	100.4	7,155	2017-10-09	42955	2017-061A
33	IRIDIUM 134	783.1	786.5	86.4	100.4	7,155	2017-12-23	43075	2017-083F
34	IRIDIUM 135	783.2	786.4	86.4	100.4	7,155	2017-12-23	43070	2017-083A
35	IRIDIUM 136	783.6	786.0	86.4	100.4	7,155	2017-10-09	42962	2017-061H

Table A.1 (Continued): Satellites used in the *Constrained*-Iridium-NEXT constellation.

Sl.No.	Name	Perigee (km)	Apogee (km)	Inclination (degrees)	Period (mins)	Semi major axis (km)	Launch Date (YYYY-MM-DD)	NORAD ID	CASPAR ID
36	IRIDIUM 137	783.6	786.0	86.4	100.4	7,155	2017-12-23	43076	2017-083G
37	IRIDIUM 138	783.4	786.3	86.4	100.4	7,155	2017-12-23	43071	2017-083B
38	IRIDIUM 139	783.4	786.2	86.4	100.4	7,155	2017-10-09	42963	2017-061J
39	IRIDIUM 140	782.5	787.1	86.4	100.4	7,155	2018-03-30	43252	2018-030F
40	IRIDIUM 141	783.5	786.1	86.4	100.4	7,155	2017-12-23	43077	2017-083H
41	IRIDIUM 142	783.3	786.3	86.4	100.4	7,155	2018-03-30	43256	2018-030J
42	IRIDIUM 143	783.0	786.6	86.4	100.4	7,155	2018-03-30	43258	2018-030C
43	IRIDIUM 144	783.0	786.6	86.4	100.4	7,155	2018-03-30	43249	2018-030A
44	IRIDIUM 145	782.7	787.0	86.4	100.4	7,155	2018-03-30	43253	2018-030G
45	IRIDIUM 146	783.2	786.4	86.4	100.4	7,155	2018-03-30	43254	2018-030H
46	IRIDIUM 147	783.6	786.0	86.4	100.4	7,155	2018-05-22	43480	2018-047E
47	IRIDIUM 148	783.5	786.2	86.4	100.4	7,155	2018-03-30	43255	2018-030I
48	IRIDIUM 149	783.0	786.6	86.4	100.4	7,155	2018-03-30	43250	2018-030D
49	IRIDIUM 150	783.1	786.5	86.4	100.4	7,155	2018-03-30	43257	2018-030B
50	IRIDIUM 151	783.6	786.0	86.4	100.4	7,155	2017-12-23	43074	2017-083E
51	IRIDIUM 152	783.4	786.2	86.4	100.4	7,155	2018-05-22	43479	2018-047D
52	IRIDIUM 153	783.4	786.2	86.4	100.4	7,155	2017-12-23	43078	2017-083J
53	IRIDIUM 154	783.6	786.0	86.4	100.4	7,155	2018-07-25	43574	2018-061F
54	IRIDIUM 155	783.1	786.5	86.4	100.4	7,155	2018-07-25	43573	2018-061E
55	IRIDIUM 156	783.1	786.5	86.4	100.4	7,155	2018-07-25	43576	2018-061H
56	IRIDIUM 157	783.5	786.1	86.4	100.4	7,155	2018-03-30	43251	2018-030E
57	IRIDIUM 158	783.3	786.3	86.4	100.4	7,155	2018-07-25	43571	2018-061C
58	IRIDIUM 159	783.5	786.1	86.4	100.4	7,155	2018-07-25	43578	2018-061K
59	IRIDIUM 160	783.6	786.0	86.4	100.4	7,155	2018-07-25	43569	2018-061A
60	IRIDIUM 161	753.3	756.4	86.5	99.8	7,125	2018-05-22	43478	2018-047C
61	IRIDIUM 162	753.1	756.6	86.5	99.8	7,125	2018-05-22	43482	2018-047G
62	IRIDIUM 163	783.2	786.4	86.4	100.4	7,155	2018-07-25	43575	2018-061G
63	IRIDIUM 164	783.5	786.1	86.4	100.4	7,155	2018-07-25	43577	2018-061J
64	IRIDIUM 165	783.3	786.3	86.4	100.4	7,155	2018-07-25	43572	2018-061D
65	IRIDIUM 166	783.2	786.4	86.4	100.4	7,155	2018-07-25	43570	2018-061B
66	IRIDIUM 167	783.9	785.8	86.4	100.4	7,155	2019-01-11	43931	2019-002K
67	IRIDIUM 168	783.6	786.0	86.4	100.4	7,155	2019-01-11	43924	2019-002C
68	IRIDIUM 169	766.3	774.2	86.4	100.1	7,141	2019-01-11	43926	2019-002E
69	IRIDIUM 170	659.5	662.6	86.6	97.8	7,032	2019-01-11	43930	2019-002J
70	IRIDIUM 171	783.6	786.0	86.4	100.4	7,155	2019-01-11	43929	2019-002H
71	IRIDIUM 172	783.5	786.1	86.4	100.4	7,155	2019-01-11	43927	2019-002F
72	IRIDIUM 173	783.5	786.1	86.4	100.4	7,155	2019-01-11	43925	2019-002D
73	IRIDIUM 175	659.0	663.1	86.6	97.8	7,032	2019-01-11	43928	2019-002G
74	IRIDIUM 176	659.6	662.5	86.6	97.8	7,032	2019-01-11	43923	2019-002B
75	IRIDIUM 180	783.8	785.8	86.4	100.4	7,155	2019-01-11	43922	2019-002A

Table A.2: Satellites/Rocket Bodies (R/Bs) used in the DebriNet constellation.

Sl.No.	Name	Perigee (km)	Apogee (km)	Inclination (degrees)	Period (mins)	Semi major axis (km)	Launch Date (YYYY-MM-DD)	NORAD ID	CASPAR ID
1	PSLV R/B	265.0	20,653.3	17.8	362.2	16,830	2008-10-22	33406	2008-052B
2	PSLV R/B	574.2	606.1	98.2	96.3	6,961	2010-07-12	36800	2010-035F
3	PSLV R/B	794.0	821.9	98.3	100.9	7,178	2011-04-20	37390	2011-015D
4	PSLV R/B	282.4	19,976.9	17.9	351.6	16,500	2011-07-15	37747	2011-034B
5	PSLV R/B	785.7	871.7	19.8	101.3	7,199	2011-10-12	37842	2011-058E
6	PSLV R/B	639.1	648.4	98.1	97.4	7,014	2012-09-09	38757	2012-047C
7	PSLV R/B	710.3	775.9	98.3	99.5	7,114	2013-02-25	39093	2013-009H
8	PSLV R/B	255.4	19,677.8	17.8	346.4	16,337	2013-07-01	39200	2013-034B
9	PSLV R/B	280.3	22,555.9	19.4	393.5	17,789	2013-11-05	39371	2013-060B
10	PSLV R/B	304.5	19,901.7	19.0	350.7	16,474	2014-04-04	39636	2014-017B
11	PSLV R/B	624.7	656.4	98.2	97.4	7,011	2014-06-30	40058	2014-034F
12	PSLV R/B	287.4	19,861.3	17.8	349.8	16,445	2014-10-15	40270	2014-061B
13	PSLV R/B	243.6	20,134.8	19.0	353.5	16,560	2015-03-28	40548	2015-018B
14	PSLV R/B	632.6	639.9	98.0	97.3	7,007	2015-07-10	40720	2015-032F
15	PSLV R/B	612.5	654.9	6.0	97.2	7,004	2015-09-28	40937	2015-052H
16	PSLV R/B	389.1	509.1	14.7	93.4	6,820	2015-12-16	41172	2015-077G
17	PSLV R/B	392.5	20,698.2	19.2	364.9	16,916	2016-01-20	41242	2016-003B
18	PSLV R/B	262.3	19,952.4	17.8	350.9	16,478	2016-03-10	41385	2016-015B
19	PSLV R/B	311.0	20,522.0	17.8	360.8	16,787	2016-04-28	41470	2016-027B
20	PSLV R/B	437.7	465.1	97.5	93.5	6,822	2016-06-22	41620	2016-040X
21	PSLV R/B	667.2	727.1	98.1	98.6	7,068	2016-09-26	41791	2016-059J
22	PSLV R/B	819.4	822.6	98.7	101.2	7,192	2016-12-07	41878	2016-074B
23	PSLV R/B	477.9	492.9	97.4	94.2	6,856	2017-02-15	42052	2017-008DJ
24	PSLV R/B	226.7	20,201.4	19.0	354.3	16,585	2018-04-11	43287	2018-035B
25	PSLV R/B	579.0	625.0	97.6	96.6	6,973	2018-09-16	43620	2018-071C
26	PSLV R/B	419.5	484.7	97.4	93.5	6,823	2018-11-29	43739	2018-096W
27	PSLV R/B	439.5	518.1	97.4	94.0	6,849	2019-04-01	44104	2019-018AC
28	PSLV R/B	484.7	563.8	37.0	95.0	6,895	2019-05-22	44234	2019-028B
29	PSLV R/B	501.2	569.8	97.5	95.2	6,906	2019-11-27	44805	2019-081B
30	PSLV R/B	526.3	577.2	37.0	95.5	6,922	2019-12-11	44862	2019-089L

Index

G

H

I

L

M

N

O

P

R

S

T

U

V

W

Curriculum Vitae

Personal Details

- *Name*: Sarath Babu
- *Address*: Pratheeksha, Poorna Auditorium Road
  Kalady P. O., Ernakulam District
  Kerala, India 683574
- *Email*: 4sarathbabu@gmail.com, sarath.babu.2014@ieee.org
- *Web-page*: https://4sarathbabu.github.io/

Education

- Doctor of Philosophy (Ph.D.) Feb 2014 – May 2021
  Indian Institute of Space Science and Technology, Thiruvananthapuram, India.
  Specialization: Disruption Tolerant Networks and Software Defined Networks

- Master of Technology (M.Tech.) Jul 2009 – May 2011
  National Institute of Technology, Calicut, India. CGPA: 8.97/10.0
  Specialization: Computer Science & Engineering (Information Security)

- Bachelor of Technology (B.Tech.) Aug 2005 – Aug 2009
  Mahatma Gandhi University, Kottayam, India Percentage: 82.28
  Specialization: *Information Technology*

List of Other Related Publications

Journals

1. A. Chakraborty, **Sarath Babu**, and B. S. Manoj, "On achieving capacity-enhanced small-world networks," *Physica A: Statistical Mechanics and its Applications*, vol. 556, p. 124729, Oct. 2020. DOI: 10.1016/j.physa.2020.124729
2. P. Koshy, **Sarath Babu**, and B. S. Manoj, "Sliding window blockchain architecture for Internet of Things," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3338–3348, Apr. 2020. DOI: 10.1109/JIOT.2020.2967119
3. N. Anand, **Sarath Babu**, and B. S. Manoj, "On detecting compromised controller in software defined networks," *Computer Networks*, vol. 137, pp. 107–118, Jun. 2018. DOI: 10.1016/j.comnet.2018.03.021
4. D. S. Yadav, **Sarath Babu**, and B. S. Manoj, "Quasi path restoration: A post-failure recovery scheme over pre-allocated backup resource for elastic optical networks," *Optical Fiber Technology*, vol. 41, pp. 139–154, Mar. 2018. DOI: 10.1016/j.yofte.2018.01.011

Conferences

1. D. Dalai, **Sarath Babu**, and B. S. Manoj, "On using edge servers in 5G satellite networks," in *Proceedings of the 3rd IEEE 5G World Forum (5GWF)*, Sep. 2020, pp. 553–558. DOI: 10.1109/5GWF49715.2020.9221366

2. R. Suraj, **Sarath Babu**, D. Dalai, and B. S. Manoj, "DebriNet: An opportunistic software defined networking framework over PSLV debris," in *Proceedings of the IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, Dec. 2019, pp. 1–6. DOI: 10.1109/ANTS47819.2019.9118082

3. T. Abhiroop, **Sarath Babu**, and B. S. Manoj, "A machine learning approach for detecting DoS attacks in SDN switches," in *Proceedings of the 24th National Conference on Communications (NCC)*, Feb. 2018, pp. 1–6. DOI: 10.1109/NCC.2018.8600196

4. P. V. Mithun, **Sarath Babu**, and B. S. Manoj, "On resolving network view inconsistencies in SDN control plane," in *Proceedings of the IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, Dec. 2017, pp. 1–6. DOI: 10.1109/ANTS.2017.8384108

5. G. Gupta, **Sarath Babu**, and B. S. Manoj, "Dual-mode TCP: An alternative approach for delay tolerant networks," in *Proceedings of the 23rd National Conference on Communications (NCC)*, Mar. 2017, pp. 1–6. DOI: 10.1109/NCC.2017.8077040

6. R. Raj, **Sarath Babu**, K. Benson, G. Jain, B. S. Manoj, and N. Venkatasubramanian, "Efficient path rescheduling of heterogeneous mobile data collectors for dynamic events in shanty town emergency response," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, Dec. 2015, pp. 1–7. DOI: 10.1109/GLOCOM.2015.7417610

7. A. V. Mamidi, **Sarath Babu**, and B. S. Manoj, "Dynamic multi-hop switch handoffs in software defined wireless mesh networks," in *2015 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, Dec. 2015, pp. 1–6. DOI: 10.1109/ANTS.2015.7413638

8. G. Jain, **Sarath Babu**, R. Raj, K. Benson, B. S. Manoj, and N. Venkatasubramanian, "On disaster information gathering in a complex shanty town terrain," in *2014 IEEE Global Humanitarian Technology Conference - South Asia Satellite (GHTC-SAS)*, Sep. 2014, pp. 147–153. DOI: 10.1109/GHTC-SAS.2014.6967574

ArXiv Preprint

1. **Sarath Babu**, G. Jain, and B. S. Manoj, "Urban Delay Tolerant Network Simulator (UDTNSim v0.1)," *CoRR*, vol. abs/1709.05645, Sep. 2017. Online: http://arxiv.org/abs/1709.05645

Software Developed

1. **Software Defined Disruption Tolerant Networking (SD-DTN) switch**

2. **Software Defined Optimized Link State Routing (SD-OLSR) protocol**

3. **Urban Delay Tolerant Network Simulator (UDTNSim)**
   [Available at: https://github.com/4sarathbabu/UDTNSim]

Doctoral Committee Members

Chairman: **Dr. N. Selvaganesan**
Professor
Department of Avionics
Indian Institute of Space Science and Technology, Thiruvananthapuram

Convenor: **Dr. B. S. Manoj**
Professor
Department of Avionics
Indian Institute of Space Science and Technology, Thiruvananthapuram

Members: **Dr. Venkata Ramana Badarla**
Associate Professor & Head
Department of Computer Science & Engineering
Indian Institute of Technology, Tirupati

Dr. Priya Chandran
Professor
Department of Computer Science & Engineering
National Institute of Technology, Calicut

Dr. Elizabeth Sherly
Director & Professor
Indian Institute of Information Technology & Management, Kerala

Dr. Rajeevan P. P.
Associate Professor
Department of Avionics
Indian Institute of Space Science and Technology, Thiruvananthapuram

Dr. Anil Kumar C. V.
Professor
Department of Mathematics
Indian Institute of Space Science and Technology, Thiruvananthapuram

Dr. Sumitra S.
Associate Professor
Department of Mathematics
Indian Institute of Space Science and Technology, Thiruvananthapuram