Improving Transport Layer Performance over Multi-hop Wireless Networks by Machine Learning

by

Nasim Arianpoo

B.Sc., University of Tehran, Tehran, Iran, 2005
M.Sc., The University of British Columbia, Vancouver, Canada, 2008

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

The Faculty of Graduate and Postdoctoral Studies

(Electrical and Computer Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA
(Vancouver)

April 2017

Abstract

The emerging use of multi-homed wireless devices along with simultaneous multi-path data transfer offers tremendous potentials to improve the capacity of multi-hop wireless networks. Concurrent Multi-path Transfer (CMT) over Stream Control Transmission Protocol (SCTP) is a form of reliable multi-path transport layer protocol with unique features that resonate with multi-path nature of the multi-hop wireless networks. The present thesis identifies and addresses some of the challenges of CMT-SCTP over wireless multi-hop networks.

One main challenge raised by the multi-hop wireless network for CMT-SCTP is the out-of-order packet arrival. SCTP uses packet sequence number to ensure delivery. As such, the out-of-order packet arrival triggers receive buffer blocking in CMT-SCTP that causes throughput degradation. Another challenge in using CMT-SCTP over multi-hop wireless networks is the unfair resource allocation towards flows coming from farther away hops.

The first part of this thesis focuses on integrating machine learning and network coding in CMT-SCTP to resolve the receive buffer blocking problem. Our state-of-the-art scheme uses Q-learning, a form of Reinforcement Learning (RL), to enable the network coding module to adjust to network dynamics. We confirm the veracity of our proposal by a queuing theory based mathematical model. Moreover, the effectiveness of the proposed scheme is demonstrated through simulations and testbed experiments.

In the second part of the thesis, we investigate the fairness behavior of CMT-SCTP

towards other CMT or non-CMT flows coming from farther away hops on a multi-hop wireless network. We introduce a Q-learning distributed mechanism to enhance fairness in CMT-SCTP. The proposed method uses Q-learning to acquire knowledge about the dynamics of the network. Consequently, the acquired knowledge is used to choose the best action to improve the fairness index of the network. We evaluate our proposal against standard CMT-SCTP and Resource Pool CMT-SCTP (CMT/RP-SCTP).

In the third part of this thesis, we apply our findings in the second part to TCP to demonstrate that the benefits of our fairness mechanism can be extended to other transport layer protocols. The findings of this thesis bring us closer to realization of the vast potential of multi-path data transfer over multi-hop wireless networks.

Preface

Hereby, I declare that the present thesis is my own original work. The work in Chapters 2 - 5 is published. The corresponding papers are co-authored by Dr. Victor C.M. Leung who supervised me through this research. The papers corresponding to Chapter 3 are co-authored by Dr Aydin who has contributed to the work by providing the QualNet module for CMT-SCTP simulations. Moreover, the papers corresponding to Chapter 5 are co-authored by Dr. Jokar who has contributed to the protocol stack design of the proposal.

Journal Papers, Published

- Nasim Arianpoo, Paria Jokar, Victor C.M Leung, "Applications of network coding to improve TCP performance over wireless mesh networks: a survey'', *Wireless Communications and Mobile Computing*, 2014 (Chapter 1)

- N. Arianpoo, I. Aydin, V. Leung, "Network Coding As a Performance Booster for Concurrent Multi-path Transfer of Data In Multi-hop Wireless Networks," in *IEEE Transactions on Mobile Computing*, vol.PP, no.99, pp.1-1 (Chapter 3)

- Nasim Arianpoo, Victor C.M Leung, "A smart fairness mechanism for Concurrent multipath transfer in SCTP over wireless multi-hop networks ", in *Ad Hoc Networks, Volume 55, February 2017, Pages 40-49* (Chapter 4)

- Nasim Arianpoo, Victor C.M Leung, "How network monitoring and reinforcement learning can improve tcp fairness in wireless multi-hop networks", in *EURASIP Journal on Wireless Communications and Networking 2016* (Chapter 5)

Conference Papers, Published

- Nasim Arianpoo, Paria Jokar, Victor C.M Leung, "Enhancing TCP performance in wireless mesh networks by cross layer design", *Conference on Computing, Networking and Communications (ICNC), 2012 International*, 2012 (Chapter 5)

- Nasim Arianpoo, Ilknur Aydini, Victor C.M Leung, "Network coding: A remedy for receive buffer blocking in the concurrent multipath transfer of data over multi-hop wireless networks", *IEEE International Conference on Communications (ICC 2014)*, 2014 (Chapter 3)

Table of Contents

Abstract ii

Preface iv

Table of Contents vi

List of Tables x

List of Figures xi

List of Abbreviations xiv

Acknowledgments xvi

Dedication xvii

1 Introduction 1

1.1 Overview of CMT-SCTP 5

1.2 Challenges of CMT-SCTP over Wireless Multi-hop Networks 6

1.2.1 Receive buffer blocking 6

1.2.2 Fairness 7

1.3 Summary of Results and Contributions 14

1.4 Thesis Organization 16

2 Preliminaries . . . 17
2.1 Network Coding . . . 17
2.1.1 Why network coding? . . . 17
2.1.2 What is network coding? . . . 18
2.1.3 How to estimate the number of redundant packets? . . . 19
2.2 Markov Decision Process . . . 20
2.2.1 What is MDP? . . . 20
2.2.2 Why MDP is a good fit for multi-hop wireless environment? . . . 20
2.2.3 Q-learning . . . 21
2.3 Testbed Setup . . . 23
2.3.1 CMT-SCTP testbed setup . . . 24
2.3.2 TCP testbed setup . . . 25

3 Receive Buffer in CMT-SCTP . . . 26
3.1 Introduction . . . 26
3.2 Coded CMT-SCTP . . . 27
3.2.1 Sender side . . . 28
3.2.2 Receiver side . . . 32
3.3 A Queuing Theory Based Proof on the Performance Gain of Coded CMT-SCTP . . . 34
3.3.1 Virtual queue in the coded multi-path transport layer . . . 35
3.3.2 Re-sequencing queue in standard multi-path transport layer . . . 36
3.4 Testbed Experiments For Accuracy Analysis Of The Q-learning Algorithm 39
3.5 Simulations For The Performance Of Coded CMT-SCTP . . . 42
3.6 Testbed Experiments For Performance Of Coded CMT-SCTP . . . 46
3.7 Discussion . . . 49

3.8 Conclusion 50

4 Distributed Fairness For CMT-SCTP 51

4.1 Introduction 51

4.2 Study of CMT-SCTP Fairness 52

4.3 Q-learning CMT-SCTP 58

4.3.1 Features and states 58

4.3.2 Actions 59

4.3.3 Reward function 60

4.3.4 Architecture 61

4.4 Evaluation 63

4.5 Discussion and Comparison 66

4.6 Conclusion 69

5 TCP Fairness over Wireless Multi-hop Network 71

5.1 Q-learning TCP Architecture 72

5.1.1 States 73

5.1.2 Action set 75

5.1.3 Transition probabilities 76

5.1.4 Reward function 76

5.2 Integration of Q-learning Agent and TCP 77

5.3 Performance Evaluations 78

5.3.1 Chain topology 79

5.3.2 Larger scale WMN 83

5.3.3 Testbed 86

5.4 Discussion and Comparison 88

5.5 Conclusion 90

6 Conclusions and Future Work 92

6.1 Research Contributions 92

6.2 Suggestions For Future Work 94

Bibliography 96

List of Tables

3.1 Performance comparison between naive Bayes and logistic regression classifiers. 42

4.1 Throughput comparison of different scenarios. In scenario 1 of Figure 4.1, nodes A and B send data using TCP. In scenario 2, all sources except node B use TCP; node B uses SCTP. Scenario 3 is similar to scenario 2; however, node B uses CMT-SCTP. In scenario 4, all nodes use SCTP. Scenario 5, all nodes use SCTP, node B uses CMT-SCTP. All the above numbers are measured in kbps. 55

4.2 Evaluation scenarios 64

4.3 Brief comparison on different flavors of CMT-SCTP 69

5.1 Network metrics parameters for different TCP variations of Figure 5.1 . . . 83

5.2 Network metrics parameters for different TCP variations of scenario 5.7 . . 86

5.3 A comparison between Q-learning TCP and TCP Reno 88

5.4 A comparison between Q-learning TCP and other well-known TCP solutions. The throughput and fairness enhancement of each TCP flavor is compared against the standard TCP. As such, when a decrease/increase is mentioned, it is relative to standard TCP. 91

List of Figures

2.1 Testbed setup for CMT-SCTP evaluations 24

2.2 IEEE 802.11 channels 25

2.3 Testbed setup for TCP evaluations 25

3.1 Markov Decision Process of coded CMT-SCTP. When the transition probabilities $P_1, P_2, ..., P_6$ are unknown, Q-learning is used to learn the reward on each state-action pair. 30

3.2 Markov chain for the length of virtual queue in coded CMT-SCTP. 36

3.3 Queuing schematic for the re-sequencing process in standard multi-path data transfer; server 1 models the channel behavior while server 2 models the re-sequencing behavior of receive buffer. 37

3.4 Markov chain for the re-sequencing behavior of the receive buffer. 37

3.5 Testbed topology - showing multiple data paths for CMT-SCTP. 39

3.6 Simulation topology - Actual data flow is over chain A, while interference traffic is over chain B. 42

3.7 Goodput of standard CMT-SCTP vs. coded CMT-SCTP (proposed in this chapter) vs. coded CMT-SCTP ver. 1 [1] with different receive buffer (rBuf) sizes, in simulation. 45

3.8 Percentage of packets in CMT-SCTP association that experience receive buffer blocking for standard CMT-SCTP vs. coded CMT-SCTP. 47

3.9 Goodput of the standard CMT-SCTP vs. CMT-SCTP with buffer splitting [2] vs. coded CMT-SCTP with different rBuf sizes, over the testbed. 48

4.1 Testbed topology. 53

4.2 Fairness comparison of different scenarios. In scenario 1, node A and B in Figure 4.1 send data using TCP. In scenario 2, all sources except node B use TCP; node B uses SCTP. Scenario 3 is similar to scenario 2; however, node B uses CMT-SCTP. In scenario 4, all nodes use SCTP. Scenario 5 is similar to 4 except for node B that uses CMT-SCTP. 56

4.3 Comparison of the average congestion window size of node B in scenario 2 and 3 of Figure 4.1. The solid line shows the average window size of node B when all sources are using TCP and node B used CMT-SCTP; while the dashed line shows the average window size of node B when all source nodes are using TCP and node B uses SCTP. 57

4.4 The MDP of CMT-SCTP Q-learning agent. The learning agent can choose one of the actions in each state based on the Q-learning rule. 59

4.5 The Jain's fairness index of Q-learning SCTP and CMT-SCTP in comparison with standard SCTP and CMT-SCTP in different scenarios of Table 4.2 64

4.6 Average throughput of different scenarios (kbps) in different scenarios of Table 4.2. Scenarios 1,2,3 and 4 are the standard SCTP and CMT-SCTP, while scenarios 5, 6, 7, and 8 are the Q-learning SCTP and CMT-SCTP. Scenario 9 shows the measurement results of CMT/RP-SCTP. 65

4.7 Average throughput of flows with different level of QoS using Q-learning fairness mechanism. 68

5.1 Chain topology - 3 nodes, 2 flows 79

5.2 Throughput changes of the flows in the chain topology of Figure 5.1. Each cycle is equivalent of 40 seconds 80
5.3 Learning rate (proof of convergence) in topology of Figure 5.1. Each learning cycle consists of 40 seconds. 81
5.4 Network utilization factor (kbytes/sec) for the topology shown in Figure 5.1. Each learning cycle consists of 40 seconds. 82
5.5 Jain's fairness index in the chain topology of Figure 5.1. Each cycle is equivalent of 40 seconds 83
5.6 Performance comparison between legacy TCP, Q-learning TCP, TCP-AP, and TCP ex Machina 84
5.7 Random network topology - 10 nodes 84
5.8 TCP flow throughput and network utility in kbytes/sec for scenario of Figure 5.7 85
5.9 Learning rate of nodes 3, 5, and 7 in scenario of Figure 5.7 86
5.10 Testbed topology. 87

List of Abbreviations

ACK	Acknowledgment
CMT	Concurrent Multi-path Transfer
ECN	Explicit Congestion Notification
IETF	Internet Engineering Task Force
MAC	Medium Access Control
MDP	Markov Decision Process
MHHP	More Hops Higher Priority
NRED	Neighborhood RED
OSI	Open Systems Interconnection
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RED	Random Early Detection
RLC	Random Linear Network coding
RP-SCTP	Resource Pool SCTP
RRSI	Received Signal Strength Indication
RTSCTS	Request-to-Send/Clear-to-Send
RTT	Round Trip Time
SACK	Selective Acknowledgment
SCTP	Stream Control Transmission Protocol

TCP	Transmission Control Protocol
TSN	Transmission Sequence Numbers

Acknowledgments

I would like to express my deepest sense of gratitude to my supervisor Dr. Victor C.M. Leung without whom this research would not be possible. I want to thank him for his continuous support, patience, and technical support.

I am also thankful to my best friend and companion Ali Tajsekandar for his unconditional love and support both technical and non-technical without whom this journey would not be possible. I am always grateful for his support and understanding during the whole time.

I feel indebted to my great family, my father, my mother and my beloved sister, Nastaran for their everlasting love, support, and protectivity not only in years of doing PhD but also throughout my entire life. Without their love and support I was not where I am today now.

This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

Dedication

To Mom & Dad

Chapter 1

Introduction

Multi-hop wireless networks have gained attention in recent years as they can extend the network coverage via providing non-line-of-sight connectivity. Moreover, the low deployment costs of these networks, as compared to for example cellular single-hop technologies such as cellular in which the coverage area depends on the number of base stations, makes multi-hop wireless networks an attractive solution for the last mile Internet access [3]. However, multi-hop wireless networks also have certain disadvantages as well. The present thesis aims to mitigate some of these disadvantages, and offers the possibility of making multi-hop wireless networks a more attractive solution for last mile access.

Multi-hop wireless networks operate based on the idea of cooperation among the nodes for packet forwarding. Multi-hop wireless networks exist in different sizes and shapes; it can be a small residential network or a network in a large commercial complex. When the destination is out of the transmission range of the sender, intermediate nodes act as relays and forward the packet to the destination. Therefore, a packet may pass via multiple hops before reaching its final destination. However, the multi-hop nature of these networks makes them vulnerable towards link failure or variable link quality which may result in dynamic/frequent data path changes. This drastically degrades the transport layer protocol performance [4]. The objective of the transport layer protocol is to provide an end-to-end connection between the peers and to relieve the upper layer from the concerns and issues of the lower layers. The Transmission Control Protocol (TCP) is the oldest available reliable transport layer protocol that is widely used over the network. TCP was

designed for and does well in a wired environment. TCP relies on the smooth changes in Round Trip Time (RTT) of packets, but, the dynamic nature of the paths in a multi-hop wireless environment can cause substantial and sudden changes in RTT. Moreover, the lack of knowledge on the available resources of the shared medium, i.e., the wireless channel, and the dynamic network topology violates the design principles of TCP. Hence, TCP fails to perform well over multi-hop wireless settings. Stream Control Transmission Protocol (SCTP), another connection oriented transport layer protocol, shares similarities with TCP; yet, SCTP addresses some of the short comings of TCP for newer applications. The unique built-in features of SCTP, such as multi-homing and multi-streaming, makes it an attractive choice as a reliable multi-path transport layer protocol in data networks. *Multi-homing* is the ability of an end-point to support more than one IP addresses. *Multi-streaming* is the ability of the transport layer protocol to segment and transmit the data into parallel streams. Multi-homing and multi-streaming position SCTP as a promising candidate for newer applications over multi-hop wireless networks. Newer applications use parallel streams to transmit data to maximize the network utilization. This feature resonates with the multipath nature of the multi-hop wireless networks. However, without a proper transport layer protocol that can support multi-path transfer, the great potentials of newer applications cannot be realized completely over multi-hop wireless networks.

At present, two main approaches account for reliable multi-path transport layer protocols: the Multi-Path TCP (MPTCP) extension for Transmission Control Protocol (TCP) [5], and the Concurrent Multi-path Transfer (CMT) extension for Stream Control Transmission Protocol (SCTP) [6]. CMT-SCTP is a more attractive approach for multi-path data transfer, since SCTP already supports multi-homing and the work required to address the shortcomings of CMT-SCTP over multi-hop wireless environment is minimal as compared to MPTCP. CMT-SCTP [7][8] is one of the first proposals to incorporate con-

current transfer of data over multiple disjoint sub-flows into the original SCTP. The main objective of CMT-SCTP is to increase network utilization and resource usage via parallel data transfer over different paths between the sender and receiver.

The multi-path data transfer in CMT-SCTP offers great potentials to multi-hop wireless networks in terms of resource usage and reliability. However, there are some limitations of using CMT-SCTP over multi-hop wireless settings:

- One of the key challenges of designing a multi-path transport layer protocol over wireless multi-hop networks is the *receive buffer blocking* caused by the out-of-order packet arrivals [8][9][10] [11]. The receive buffer stores out-of-order packets and delivers them to the application layer consequently after having received all the missing packets. Due to path discrepancies in multi-path data transfer, the receive buffer can be flooded by out-of-order packets. As a result, the data sender can be throttled, which causes the overall throughput of the connection to be reduced or even become zero at times. This phenomenon is called *receive buffer blocking*. Receive buffer blocking results in performance degradation at the transport layer. The receive buffer blocking exacerbates over wireless multi-hop networks as the unreliability of the paths creates more complexity for the transport layer. Of the existing literature, that focuses on addressing receive buffer blocking over simplified scenarios, none of the methods have been tested over multi-hop wireless networks [10][11] [12][13][8][9].

- The fairness behavior of a multi-path transport protocol is another major challenge when deployed in wireless multi-hop networks. It can inflict harm not only to the protocol itself but also to the performance of other existing protocols such as TCP flows or other single-homed SCTP flows [14][15][16][17]. The fairness of CMT-SCTP has been partially investigated over simple bottleneck scenarios [18] and [19]; however, studies that have focused on fairness behavior of CMT-SCTP over wireless multi-hop

networks against non-CMT flows coming from farther away hops are limited.

A significant portion of the proposed algorithms have focused on wired settings and the resulting insights cannot be applied to wireless multi-hop networks [20] [21] [22] [23][24] [25][9] [10] [26][8] [27]. The wireless multi-hop setting introduces both severe path dissimilarities and severe unfairness to the picture for CMT-SCTP. Therefore, novel proposals are required to accommodate the intrinsic characteristics of the wireless multi-hop network within the CMT-SCTP. We focus our interest on the performance of CMT-SCTP over a wireless multi-hop setting to address the shortcomings of the literature available.

The main questions of this thesis are (1) how to address the fairness issue of the multi-path transport layer protocol over multi-hop wireless networks and (2) how to enhance performance of the multi-path transport layer over multi-hop wireless settings by alleviating receive buffer blocking.

The present thesis is divided into two main parts: the first part focuses on algorithms to alleviate receive buffer blocking in CMT-SCTP over wireless multi-hop networks, whereas the second part focuses on improving CMT-SCTP fairness against non-CMT flows coming from farther away hops within the wireless multi-hop setting. The results provide insight into the behavior of CMT-SCTP over wireless multi-hop networks which can be used to improve the performance of CMT-SCTP over the existing infrastructure. The proposals in the present thesis are one step ahead in transiting swiftly from legacy transport protocols to multi-path transport protocols. In addition, the requirements of newer applications are addressed.

The remainder of this chapter is organized as follows: In Section 1.1 we highlight details of CMT-SCTP and its known issues. Subsequently, we identify the challenges of using CMT-SCTP over wireless multi-hop networks and explain how addressing these challenges may contribute to the overall goal of this thesis, which is the focus of Section

1.2. In Section 1.3, we focus on the mechanisms proposed in this thesis, and explain the contribution and novelty of the solutions. In Section 1.4, a detailed map of the thesis is presented.

1.1 Overview of CMT-SCTP

Except for some minor differences, SCTP and TCP share the same congestion control mechanism. TCP uses Acknowledgment packets (ACK) to inform the transmitter about the missing packets using the packets' sequence numbers. However, SCTP uses Selective Acknowledgment (SACK) to acknowledge both the received data and the gaps in the received data. Moreover, TCP is byte oriented, whereas SCTP is message (chunk) oriented. Aside from the minor differences in the congestion control mechanism of SCTP and TCP, multi-homing and multi-streaming are unique to SCTP. In multi-homing, SCTP socket holds more than one IP addresses for each host and creates redundancy for transport layer. Therefore, should any of the interfaces fails, the SCTP association continues to transmit data on the other interface. In multi-streaming, the SCTP association divides the data into multiple streams and transmits each stream independently. SCTP association uses the Transition Sequence Number (TSN) of each packet to keep track of packet losses within each stream. The association uses the stream identifier/stream sequence number to track any gaps in data delivery to the application layer. When there is a missing packet in one of the streams in a multi-stream association, data chunks from unaffected streams can be delivered to the upper layer. The main objective of multi-streaming in SCTP is to increase redundancy and resilience against network failure. Multi-streaming in its original form does not support simultaneous data transmission over all streams. Therefore, Concurrent Multi-path Transfer of data was proposed as an adds-on suite to SCTP [8] . CMT-SCTP uses multi-homing in an SCTP association to transmit data concurrently over all available

sub-flows within the association. The main objective of CMT-SCTP is to increase network utilization and resource usage.

1.2 Challenges of CMT-SCTP over Wireless Multi-hop Networks

The wireless multi-hop environment induces unexpected characteristics that are not addressed in the original design of CMT-SCTP. Therefore, CMT-SCTP fails to enhance network utilization over multi-hop wireless networks. The un-addressed challenges create interesting research topics.

1.2.1 Receive buffer blocking

Considering the drawbacks of receive buffer blocking on the performance of the multi-path transport layer protocol, the receive buffer blocking topic is investigated by many researchers. One of the common practices is to set aside enough space in the receive buffer [12]. To decrease the possibility of receive buffer blocking during the lifetime of the transport connection in multi-path transport protocol, Barre *et al.* [13] suggested to reserve a space of twice the bandwidth delay product of the path. Although the proposed mechanism in [13] mitigates the receive buffer blocking, reserving space in the receive buffer decreases the receiver tolerance against the changes in path delays. To solve the issue of receive buffer blocking in CMT-SCTP, Iyengar *et al.* [8] proposed different retransmission policies. These policies are based on the packet loss rate and the congestion window size of each path [8]. Although Iyengar's approach of using a proper retransmission policy alleviates the receive buffer blocking problem, in the presence of non-negligible and unavoidable path dissimilarities, none of the proposed retransmission policies is effective in eliminating the receive buffer blocking problem. To improve the performance of SCTP-CMT over

dissimilar paths, Deibholz *et al.* [2] proposed a resource pooling algorithm that divides the receive buffer among all the paths to prevent the receive buffer from being flooded by out-of-order packets from one path and prevent other paths from sending data. Deibholz's resource pooling partially alleviates buffer blocking and reduces the receive buffer blocking problem from the entire buffer to a portion of the buffer. However, in the presence of severe path dissimilarities, the local blocking infiltrates to the rest of the buffer. In Chapter 3, we propose an adaptive network coding mechanism to desensitize the receiver against packet reordering and consequently eliminate the receive buffer blocking problem. Our mechanism integrates network coding [28][29] into CMT-SCTP to resolve receive buffer blocking. Our state-of-the-art network coding scheme uses a combination of Q-learning [30] and logistic regression for rare data events to control the number of redundant packets based on the network dynamics.

1.2.2 Fairness

Fairness measures have different definitions based on the perspective and can be categorized into three groups [18]:

- **Link-centric sub-flow fairness** only focuses on how the resources of a bottleneck is divided among the passing by sub-flows without taking into account of the total resource allocation to each flow. For example, if there are n sub-flows belonging to m flows passing through the bottleneck link l with a capacity of $C(l)$, assigning a bandwidth of $\frac{C(l)}{n}$ to each sub-flow is considered link-centric sub-flow fair.

- **Link-centric flow fairness** focuses on resource allocation to flows sharing a bottleneck regardless of the number of sub-flows. Assuming n sub-flows belonging to m flows passing through the bottleneck link l with a capacity of $C(l)$, assigning a bandwidth of $\frac{C(l)}{m}$ to each flow is considered link-centric flow fair. As such, in link-centric

flow fairness, each flow divides the allocated resource evenly among its sub-flows.

- **Network-centric fairness** divides the network resources evenly among all flows in the network regardless of the bottlenecks. (the present thesis perspective for fairness measure)

Whenever fairness is mentioned throughout this thesis, we are referring to network-centric fairness.

A substantial portion of the available literature of transport layer fairness over wireless multi-path networks is focused on link-centric fairness resolutions for TCP [4, 31–43]. Studies that focus on the fairness of SCTP or CMT-SCTP are limited. As TCP and SCTP share a similar congestion control mechanism, studying the available fairness solutions for TCP over wireless multi-hop settings provides insights into the fairness behavior of both SCTP and CMT-SCTP. Therefore, we focused on studying the available fairness mechanisms for TCP over wireless multi-hop networks. Subsequently, we developed a distributed fairness mechanism for SCTP and CMT-SCTP using a reinforcement learning approach. Eventually, we adopted the reinforcement learning fairness mechanism with additional changes to TCP [44][45]. Our methodology offers both backward compatibility towards TCP and enhances CMT-SCTP fairness over the multi-hop setting.

Fairness of TCP

The existing literature on TCP fairness solutions over wireless multi-hop networks can be categorized into cross-layer designs e.g., [44, 46–54], and layered proposals e.g., [55–68]. While layered proposals aim to keep the end-to-end semantic of TCP intact, the cross-layer designs use the information from different layers to adjust TCP parameters. Random Early Detection (RED) is among one of the first proposals to enhance TCP fairness over wired connections. Neighborhood RED (NRED) is a cross-layer adaptation of RED for wireless

multi-hop settings [48]. NRED is implemented in Medium Access Control (MAC) and uses channel utilization from the physical layer to calculate the probability of packet dropping. In [49], the gateway triggers Explicit Congestion Notification message (ECN) for the nearby TCP sources with a higher probability to favor the farther away flows and allow them to use more network resources. The fact that the gateway controls the ECN mechanism is one of the key limitations of the cross-layer approach of [49]. Raniwala *et al.* [50] proposed a cross-layer solution that uses the topology of the network to create a connection graph and calculate the flow rates. The heavy overhead of feedback messages caused by [50] to create the global graph of the network is undesirable. The cross-layer method of [51], which is called TCP-AP, attempts to eliminate the reliance of TCP fairness solutions on feedback messages. The approach of [51] relies on the information from the physical layer to infer the fair share of each node of network resources. The main drawback of TCP-AP is the reliance on received signal strength indication (RSSI), which is not an accurate estimate of the receiver power level. Therefore, TCP-AP relies on feedback from the receiver to function effectively. XCP, another example of cross-layer design [53], uses the feedback from routers to adjust the congestion control window size. XCP suffers from excessive overhead in presence of heavy traffics. CoDel [69] is another cross-layer example that uses active queue management on selected bottle neck links, i.e., links with large queuing delay. CoDel uses spacing in transmission times as the queue management method over bottle neck links. D^2TCP is a recent variation of TCP for on-line data intensive applications that uses a varying rate based on a deadline set by the application and the congestion condition of the network reported by ECN [54]. D^2TCP is a cross-layer approach that needs compatibility in both application layer and routing protocol to perform effectively which is a huge disadvantage. In [44], a cross-layer algorithm, More Hops Higher Priority (MHHP), is proposed that assigns higher priority to flows traversing a larger number of

hops. Although MHHP improves TCP fairness in a multi-hop environment, the design does not fully capture the dynamic nature of the wireless environment.

Unlike the cross-layer approaches, the layered solutions preserve the end-to-end semantic of Open Systems Interconnection (OSI) model. According to Gambiroza *et al.* [55], the binary back-off mechanism, Request-to-Send/Clear-to-Send signaling mechanism (RTS/CTS), and the end-to-end congestion mechanism play key roles in TCP unfairness. Gambiroza *et al.* [55] proposed a MAC layer solution in which each access point calculates the fair share of incoming and outgoing flows in terms of flow rates and broadcasts the rates to other nodes. Reference [57] is another MAC layer approach in which nodes negotiate to set up a fair transmission schedule throughout the network. The proposed methods in [55] and [57] rely on feedback messages, which significantly increases the network overhead. In [59], a layered approach is proposed that uses the TCP advertised window and delayed ACK to control flow rates of different flows. The algorithm in [59] only works in scenarios where all flows are destined to the gateway. In [60], authors proposed a mechanism that prioritizes TCP ACK packets in the reverse path to enhance TCP fairness. TCP BIC (Binary Increase Congestion) is an end-to-end layered approach that uses a combination of additive linear increase and binary search of the congestion window to ensure fairness among the flows [64]. When the congestion window is small, TCP uses the binary search algorithm to find the correct size of the increase to the congestion window to let the flow utilizes its network share. TCP CUBIC is an enhanced version of TCP BIC that uses the cube of the elapsed time from the last window reduction to control the congestion window size [65]. The TCP CUBIC approach creates independence for congestion control window from Round Trip Time (RTT). Therefore, TCP CUBIC increases fairness in favor of flows with longer RTT. TCP Veno, another instance of layered solutions to TCP [66], gained a lot of attention in recent years due to its better performance over wireless settings. Veno is

basically an integration of the idea of congestion detection of TCP Vegas into TCP Reno and does not significantly contribute to the fairness. TCP Jersey [70] is another flavor of TCP that focuses on recognizing the random losses of the wireless environment from the congestion loss. Among the end-to-end solutions to enhance TCP performance, only few use machine learning as an interactive tool to observe network dynamics and change TCP parameters based on some prediction of the network behavior. Sprout is a good example of the interactive transport layer protocol [67]. Sprout uses the arrival time of packets to predict how many bytes the sender can successfully transmit without creating congestion in the network while maximizing network utilization. The main disadvantage of Sprout is the fact that it needs to run over CoDel to outperform other TCP variants. Thus, it requires changes to the infrastructure to enable active queue management. TCP ex Machina, also known as Remy, is another end-to-end interactive solution that uses machine learning to create a congestion control algorithm, which controls the packet transmission rate based on the perceived network model by the learning algorithm [68]. The main disadvantage of TCP ex Machina is its resource-intensive nature that results in lengthy learning time. It takes almost forever for a single Linux machine to come up with a congestion algorithm suitable for a specific network using TCP ex Machina.

Fairness of CMT-SCTP

Although TCP fairness for over both wired and wireless environments has been extensively studied, SCTP fairness, specifically CMT-SCTP fairness, over wireless environment is a less explored field. One of the initial studies on CMT-SCTP performance over wireless multi-hop networks [71] focused on the throughput of CMT-SCTP over a multi-hop network in QualNet without considering the fairness issues. In another study [19], Ilknur Aydin *et al.* investigated CMT-SCTP fairness against TCP and SCTP in a wired setting and concluded that CMT-SCTP is indeed fair on wired bottleneck scenarios. However, the

superior loss-recovery mechanism in CMT-SCTP results in a better bandwidth utilization in CMT-SCTP as compared to TCP. The fairness study in [19] did not include any wireless scenarios. In a previous study that focused on CMT-SCTP fairness alone [72], it was shown that CMT-SCTP is not fair to non-CMT flows when handling the congestion control of each path independently. Consequently, authors of [72] proposed Resource Pool CMT (CMT/RP-SCTP) that integrates resource pooling into congestion control mechanism of CMT-SCTP to improve fairness against non-CMT flows. Based on [72], a CMT/RP-SCTP-based transport layer has three main objectives:

- CMT/RP-SCTP should have a throughput gain over SCTP and non-CMT flows as it is using extra paths for concurrent data transfer.
- CMT/RP-SCTP must be fair for non-CMT flows over bottlenecks.
- CMT/RP-SCTP must apply congestion control as a whole to balance the load on all the paths.

Although the above goals are satisfied in CMT/RP-SCTP over wired settings, not a single study reports on the use of CMT/RP-SCTP over a multi-hop wireless environment. Following [72], the same group of authors proposed a change in congestion control mechanism of CMT-SCTP to balance the congestion on each path while keeping it fair to single path transport protocols. The proposal used in [18] increases the congestion window size of each path proportional to the average RTT of all paths. While the fairness mechanism in [18] makes multi-path transport layer fair to single path transport layers, it does not offer Quality of Service (QoS) for costumers paying for higher bandwidths, and thus the proposal is not financially attractive to service providers. In [73], authors proposed a TCP-friendly congestion control mechanism. The method in [73] applies a Vegas-like congestion control mechanism based on the RTT variant of the paths to balance the load of each

path within the CMT-SCTP association. The algorithm in [73] is evaluated in ns-2 using a very abstract model. More detailed investigation is required to study the fairness of the proposal on competing non-CMT flows within the network.

To the extent of our knowledge, a major portion of the available literature on CMT-SCTP is focused on improving performance of CMT-SCTP as compared to SCTP [74], [75], [8], [76], [77], [78], [79]. In publications, such as [80] and [81], the performance of CMT-SCTP over wireless networks was investigated without any fairness evaluations. Studies described in [80] and [81] proposed methods to improve packet loss and quality of service of multimedia without any fairness considerations.

The importance of fairness in CMT-SCTP over wireless multi-hop networks against other non-CMT flows cannot be overemphasized. In recognition of this importance and due to the lack of extensive literature on this topic, Chapter 4 investigates the fairness of CMT-SCTP over a wireless multi-hop testbed and proposes a dynamic algorithm to improve the fairness of CMT-SCTP against non-CMT flows. Our proposal is an attempt to integrate fairness to the current CMT-SCTP design to create a more coherent network body.

In Chapter 5, we adapted our distributed fairness mechanism to TCP. In our approach, we deployed Q-learning, a reinforcement learning mechanism, to monitor the dynamics of the network. The Q-learning agent creates a state map of the network based on the MAC parameters and takes actions to enhance TCP fairness and throughput of the starved flows in the network. The proposal creates a distributed cooperative mechanism where each node hosting a TCP source uses local fairness index and aggressiveness index of the flow to adjust its TCP parameters.

1.3 Summary of Results and Contributions

This thesis contains new proposals to accommodate the intrinsic characteristics of the wireless multi-hop network within CMT-SCTP in terms of fairness and performance. The contributions of the chapters are:

- In Chapter 3, we address the issue of receive buffer blocking in CMT-SCTP. We propose a network coding scheme for CMT-SCTP that creates cooperation among the multiple sub-flows of a CMT-SCTP association to eliminate the receive buffer blocking problem. We use Random Linear Network Coding (RLC) to create cooperation among the sub-flows of a CMT-SCTP association. The cooperation among the sub-flows evenly distributes the loss rate probability on all sub-flows. Thus, the discrepancy among the sub-flows diminishes. Our contributions are as follows:
  - We integrated network coding into CMT-SCTP (called *coded CMT-SCTP*).
  - We proposed an adaptive network coding scheme to control the number of redundant packets depending on the network dynamics.
  - We proved that our scheme increases the frequency of receive buffer unloading using a queuing theory approach.
  - Our coded CMT-SCTP scheme outperforms the original CMT-SCTP by 22%-62% depending on the path dissimilarities and receive buffer size.
- In Chapter 4, we investigated the effect of utilizing more than one path on the fairness of CMT in competing with other SCTP or TCP flows coming from farther away sources. The results of our investigation motivated us to design a dynamic mechanism using reinforcement learning to alleviate the aggressive behavior of CMT-SCTP by fine tuning CMT-SCTP parameters. Note that our investigation is focused

on multi-hop networks with minimal to zero mobility, such as wireless mesh networks. The result of this study is one step forward in practical exploitation of CMT-SCTP in multi-hop home or office area networks or even in larger scale wireless networks. The contributions are as follows:

  - We evaluated the fairness of CMT-SCTP against non-CMT flows coming from farther away hops in a real multi-hop wireless environment.
  - We used our findings to develop a dynamic distributed fairness mechanism to alleviate the aggressive behavior of CMT-SCTP towards non-CMT flows coming from farther away hops.
  - We compared the results of our findings to standard CMT-SCTP and CMT/RP-SCTP. Our proposal outperforms the standard CMT-SCTP with a high margin.

- In Chapter 5, we adapted the idea of the distributed fairness mechanism to TCP. In our proposed algorithm, each node uses a reinforcement learning approach to model the dynamics of the network and fine tune TCP parameters based on the perceived characteristics of the environment. Our algorithm preserves autonomy of each node in the decision making process and does not require a central control mechanism or control message exchange among nodes. Unlike the existing machine learning solutions, our proposal is compatible with the computational capacity of the current infrastructure. We have named our approach Q-learning TCP. The contributions of Chapter 5 can be summarized as:

  - We proposed a dynamic distributed fairness mechanism for TCP over wireless multi-hop networks that does not require any feedback messages or a central control.

- We evaluated our proposal against standard TCP, TCP ex Machina and other TCP flavors over a real testbed.
- Our proposal enhances TCP fairness by 10% to 20% without any feedback messaging and any extra overhead to the medium.

1.4 Thesis Organization

The present thesis is organized as follows: In Chapter 2, we presented a brief summary of the tools we used to design our proposed algorithm along with details of our testbed setup. In Chapter 3, we proposed the coded CMT-SCTP to address the receive buffer blocking in CMT-SCTP over a multi-hop environment. The proposed mechanism is evaluated and compared against standard CMT-SCTP and proved to be effective. In Chapter 4, we proposed our distributed fairness mechanism for CMT-SCTP over multi-hop wireless networks. We evaluated our proposal against other available fairness mechanism for CMT-SCTP over multi-hop wireless setting. We used our findings in Chapter 4 to enhance TCP fairness over multi-hop wireless networks in Chapter 5. The proposed mechanism in Chapters 5 and 4 uses Q-learning to monitor the dynamics of the network and fine-tune the parameters of the transport layer to enhance fairness against flows coming from farther away hops. We used real testbeds to evaluate the performance of our proposals both in Chapter 5 and 4. Finally, this thesis is concluded in Chapter 6 and directions for future work are discussed. The findings of each chapter have been published in a journal paper.

Chapter 2

Preliminaries

In this chapter, we go over the tools we used to address the issues of the transport layer protocol over multi-hop wireless networks. Besides, we explain the details of our testbed setup. The testbed is used with different number of nodes and different topologies in each chapter; the network topology of each chapter is explained in each chapter in the testbed section.

Section 2.1 provides a brief summary on network coding. Sections 2.2 and 2.2.3 present an introduction to Markov Decision Process (MDP) and Q-learning. We go over our testbed setup in Section 2.3.

2.1 Network Coding

We use network coding in Chapter 3 to address receive buffer blocking in CMT-SCTP. But why network coding is an effective solution for addressing the receive buffer blocking?

2.1.1 Why network coding?

SCTP uses transmission sequence number of the packets as a mean to guarantee the reliable delivery of the data to the receiver. In case of a packet loss, the packets with higher transmission sequence number are stored in the receive buffer until the arrival of the missed packet. The conventional transport protocols are designed to recover from congestion losses by reducing congestion, and are inefficient when random losses are encountered. Network coding is one way to introduce redundancy in the transmitted data stream so that the

receiver needs not wait for retransmission of a dropped packet if it can be recovered from coded versions of the packet.

2.1.2 What is network coding?

Network coding is a simple but effective method in communication networks in which packets are combined together instead of simply being forwarded at the routers. The coded packets are processed to recover the original packets in the receiver. The coded packets are categorized into two groups: *redundant* and *innovative*. An innovative packet is a coded combination which is linearly independent from the previously coded packets; while a redundant packet is a coded combination with dependency on previously coded packets. The idea was first implemented by Ahlswede *et al.* [82] in a butterfly network. Following Ahlswede's work, extensive literature on practical and theoretical analysis of network coding demonstrate that network coding results in substantial improvement in both throughput and reliability of a network [83][84][28][85][86][87].

The pioneer work of Sundararajan *et al.* [86][87] in adding a network coding layer between the IP and TCP layer demonstrates that RLC can be adopted to higher layers and enhance throughput. Sundararajan *et al.* used the concept of *degrees of freedom* at the receiver side to decide whether an ACK should be sent out. If the recently arrived coded packet carries a new linear combination of the original packets (*innovative packet*), the receiver increases the number of *seen packets* so far and sends an ACK confirming a new packet arrival. The number of seen packets is a representative of the number of independent linear equations received so far. Every time a new packet is seen at the receiver, the decoding process is triggered and an original packet is recovered with a high probability. As such, network coding does not disturb the timing of the TCP acknowledgment mechanism. One of the drawbacks of the Sundararajan *et al.* approach is that the method has to be tailored to the byte-oriented nature of TCP and hence requires extra packet pre-processing.

However, the message-oriented nature of SCTP eliminates the need for pre-processing of the packets. In addition, the coding algorithm proposed in [86] is not attuned to a dynamic environment. Instead, it assumes a constant packet loss rate and transmits redundant packets just to cover the pre-calculated loss regardless of any changes in the environment which makes the proposed method less suitable for dynamic and unreliable environments such as multi-hop wireless networks.

2.1.3 How to estimate the number of redundant packets?

To recover all the original packets involved in the linear network coding process, we need to have enough linear combinations at the receiver. Thus, we need to add enough redundant packets to mask the random losses of the network. When using network coding in a dynamic environment such as multi-hop wireless network, the packet loss rate is unknown. Therefore, we need to find a way to estimate the number of redundant packets to mask the random losses of the network. We need a sequential decision making process to decide when to send an innovative and when to send a redundant packet based on the dynamics of the network. Markov Decision Process (MDP) is a powerful technique for sequential decision making problems in dynamic environments. We are going to elaborate more on MDP in 2.2.

in Chapter 3, we adapt Sundararajan's *degrees of freedom* approach to CMT-SCTP to eliminate the reliance of the congestion control algorithm on the packet's transmission sequence number. First, our approach creates cooperation among multiple sub-flows of data within an CMT-SCTP association by combining packets from different sub-flows together; this way, different characteristics of the paths are less disruptive for packet processing at the receiver side. Second, the machine learning aspect of our scheme addresses the inflexible behavior of [86] by monitoring the ever-changing nature of an underlying network and adjusting the number of redundant packets accordingly.

2.2 Markov Decision Process

In Chapters 3, 4, and 5, we use MDP [88] to model the multi-hop wireless network.

2.2.1 What is MDP?

Each MDP has finite number of states and there is a limited number of actions available to the learner module in each state. The learner module monitors the changes of environment via the feedback of the reward function. The transition probability matrix of the MDP is not known at the beginning of the learning process. One popular way to learn the transition probabilities of the states on an MDP is to use Q-learning [89].

MDP is a powerful tool to model a multi-hop wireless network. These networks have a stochastic environment that changes frequently; each node within the network is a decision maker or an agent that interacts with the environment via taking actions. The actions can be complex such as deciding whether to send a redundant or innovative packet or as simple as changing the maximum congestion window size of TCP or CMT-SCTP. The objective of the network can be translated into a reward function to give feedback to the node. The environment can be modeled as an MDP to optimize the network objective such as fairness, network utilization, or any other network objective. Besides, MDPs are very versatile and can be defined with different parameters and states to serve the design requirements of each network. For instance, in Chapter 3, we use MDP to optimize the network coding process while in Chapters 4 and 5, we use MDP to optimize the fairness aspect of the network with unique state definition in each case.

2.2.2 Why MDP is a good fit for multi-hop wireless environment?

In dynamic environment such as a multi-hop wireless network, static decision commands can lead to inefficient resource allocation. Thus the dynamic set of policies by MDP is

a logical choice. Moreover, MDP can support a balance utility function with different objectives.

When the transition probabilities of the MDP are not known, reinforcement mechanism such as Q-learning are used to infer the transition probabilities via learning. More details on Q-learning as presented in 2.2.3.

2.2.3 Q-learning

Q-learning is a class of Reinforcement Learning mechanism (RL) first introduced by Watkins *et al.* [89]. RL algorithms model the world as a Markov Decision Process (MDP). RL algorithms are based on the basic idea of learning via interaction. The learner module is called an *agent*, and the interactive object which is the subject of learning is the *environment*. During the learning process the time is divided into decision epochs. Each Q-learning agent has four elements: a set of states, a set of actions, a state-action dependent reward function, and a state-action dependent transition probability matrix.

The learning agent uses a reward function to receive feedback on the consequences of taking an action. The interaction between the agent and the environment helps the agent to construct a mapping of the possible states-actions. The agent mapping is called the *policies* and shows how the agent changes its decisions based on the different responses from the environment. As time passes, the mapping gets more inclusive and the agent learns almost all the possible $(state, action)$ pairs and their associated reward/penalty values and can cope with any changes in the environment. The memory of the agent is kept in a matrix called Q. The rows of Q represent the current state of the system and the columns represent the possible actions leading to the next state. At the beginning of the learning process, the learning agent does not know the transition probabilities of the MDP and the associated reward with each transition. After each decision epoch, Q is updated

as in equation (2.1).

$$\overbrace{Q_{t+1}(s_t,a_t)}^{new\ value} = (1-\alpha)\overbrace{Q_t(s_t,a_t)}^{old\ value} + \alpha[\underbrace{r(s_t,a_t)}_{reward\ observed\ after\ performing\ a_t\ in\ s_t} + \overbrace{\gamma \max_a Q_t(s_{t+1},a)}^{estimated\ future\ value}]^{learned\ value} \tag{2.1}$$

Equation (2.1) is called the Q-learning or Ballard rule [88]. The objective of the Q-learning rule is to inform the agent of the possible future rewards along with the immediate rewards of the latest action. α is the learning rate of the agent and determines the importance of the newly acquired reward. α varies between 0 and 1. A factor of 0 causes the agent not to learn from the latest $(action, state)$ pair, while a factor of 1 makes the agent to only consider the immediate rewards without considering the future rewards. γ is the discount factor and determines the importance of future rewards. γ varies between 0 and 1. A 0 discount factor prohibits the agent from acquiring future rewards, while a factor of 1 pushes the agent to only consider future rewards. The goal of the learning agent is to maximize its total reward via taking the suitable action in each state.

Choosing α and γ has a direct effect on the convergence of the Q-learning mechanism [88]. Based on the recommendation of [89][90] choosing $\gamma = 0.9$ and $\alpha = \frac{1}{(1+t)^\omega}$ can decrease the convergence time to a polynomial degree. Authors of [90] showed that $\omega = 0.77$ can be an optimum for the learning rate.

The *reward function* plays a key role in the Q-learning process [90][91][92]. The learning agent uses the reward function to determine the consequence of the latest action and to encourage the learning agent to stay in/approach the goal state. The reward function serves as a gradient function to the learning agent. The correct direction to maximize the learning rule is conveyed by the reward function through immediate rewards or penalties after each action. Choosing a Gaussian reward function complies with the suggestions of

[93] for a good reward function. The Gaussian function is almost uniform in states far from the goal state, and has an increasing gradient in a belt around the goal state. As such, the Gaussian reward function can provide efficient direction to the agent on how to approach the goal state from any other state.

2.3 Testbed Setup

To evaluate the performance of our proposals, we set up a testbed in an actual office environment (Figure 2.1). We used Raspberry Pies as wireless nodes in our testbed. Raspberry Pi [94] is a tiny affordable computer that can host multiple wired/wireless interfaces. In our testbed, A and B are the client (data sender) nodes, D, E, and F are data forwarder/router nodes, and G is the server (data receiver) node. We used FreeBSD [95] on client and server nodes and Debian 7 [96], another Unix-like operating system, on nodes acting as a router. While FreeBSD provides a very extensive SCTP suite, including the CMT feature, it only supports Wi-Fi adapters in the client mode, and not in the host or forwarder mode. As such, we used Debian 7 for nodes in the middle that has to play a role in forwarding/routing packets. We used TP-LINK nano USB adapters as *802.11 g* Wi-Fi interface [97], as they are cheap and they work well with FreeBSD and Debian 7.

In terms of the interference, note that there are additional sources of noise from the 40 laptop users, 5 Cisco routers, and other wireless devices (such as the cell phones of the staff) present in the office area where we set up our testbed. The data collections took place in different hours of the day to make sure that all possible outcomes are covered in our data set. We used Wireshark [98] for data collection in our measurements.

2.3.1 CMT-SCTP testbed setup

To evaluate the performance of our CMT-SCTP related proposals, we set up the testbed with the following characteristics: The testbed is set up in a way that one wireless interface of node B and C communicates with node D, which forwards data to one wireless interface of node G. The other wireless interface of node B and C communicates with node F, which forwards data to node E (using static routing). In turn, node E forwards data from node F to the other wireless interface of node G. Node A can communicate with either node D or node B depending on the testing scenario (node D for evaluations of Chapter 3 and node B for evaluations of Chapter 4), which forwards the data to one wireless interface of node G.

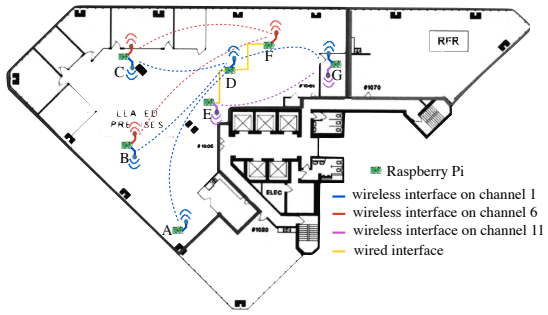


Figure 2.1: Testbed setup for CMT-SCTP evaluations

As depicted in Figure 2.1, nodes A, D, E, and F have only one wireless interface; while nodes B, C, and G have two wireless interfaces. In addition nodes D, E, and F have wired interfaces to allow static routing. Note that D has two wired interfaces. Each wireless interface is setup on a different channel (see Figure 2.2) to guarantee an independent data path for sub-flows. Only wireless interface of node A, one wireless interface of node B and C (forwarding data to node with D), only wireless interface of node D, and one wireless interface of node G (receiving data forwarded by wireless interface of node D) are tuned to channel 1; the other wireless interface of node B and C (forwarding data to node F) and

only wireless interface of node F are tuned to channel 6; only wireless interface of node E and the other wireless interface of node G (receiving data forwarded from the wireless interface of node E) are tuned to channel 11.

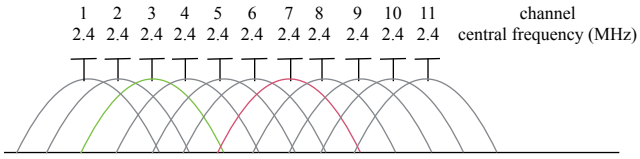


Figure 2.2: IEEE 802.11 channels

2.3.2 TCP testbed setup

To evaluate the performance of our proposal in real world setting, we set up the topology as follows: The blue nodes in Figure 2.3 are the employees with their laptops (MacBook Pro or MacBook Air) which connect to the Internet via Router R1 (extender) or R2. Routers R1 and R2 connect to the Internet through the gateway (purple node). We added node B which is both a source and forwarder node. Node B can forward to R1 and R2; moreover, we set up a static routing table inside node B that forwards packets from node A to node C without forwarding them to R1 or R2.

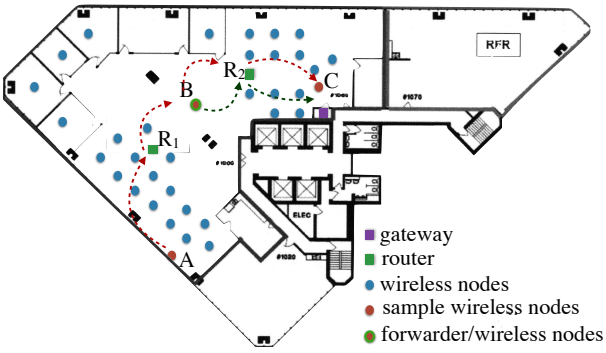


Figure 2.3: Testbed setup for TCP evaluations

Chapter 3

Network Coding As a Performance Booster for Concurrent Multi-path Transfer of Data In Multi-hop Wireless Networks

3.1 Introduction

In this chapter, we propose a network coding scheme for CMT-SCTP that creates cooperation among the multiple sub-flows of a CMT-SCTP association to eliminate the receive buffer blocking problem. The main cause of the receive buffer blocking in reliable transport layer protocols is the dependency of the congestion control mechanism on transmission sequence number of packets. The proposed scheme eliminates the reliance of transport layer protocol on TSN of packets using RLC. RLC is a technique where all participating nodes send out random linear combination of the packets in hand to utilize the lossy path capacity [29][28]. RLC uses *redundant packets* to mask the random losses of the path and to guarantee that the receiver is capable of recovering the original packets. Our proposed network coding scheme uses machine learning to control the number of redundant packets. In particular, we use a Q-learning mechanism [89](See Section 2.2.3) along with a classifier inside the Q-learning mechanism to determine when the sender has to transmit redundant packets. Our contributions are as follows:

- We integrated network coding to the multi-path transport protocol CMT-SCTP (called *coded CMT-SCTP*).

- We proposed an adaptive network coding scheme based on a Q-learning algorithm [89] to control the number of redundant packets depending on the network dynamics.
- Using a queuing theory approach, we proved that our coded CMT-SCTP scheme reduces the possibility of receive buffer blocking.
- Our coded CMT-SCTP scheme outperforms the standard CMT-SCTP with a high margin (up to 62% depending on the path dissimilarities and receive buffer size).
- We also set up a real testbed and confirmed our theoretical and simulation results.

This chapter is organized as follows. Section 3.2 presents the details of our coded CMT-SCTP scheme. Our claims in eliminating receive buffer blocking using coded CMT-SCTP is proved using queuing theory in Section 3.3. The accuracy of the machine learning mechanism is evaluated on a testbed in Section 3.4. Section 3.5 and 3.6 discuss the simulation and testbed results on performance of coded CMT-SCTP, respectively. Section 3.7 discusses how to adapt coded CMT-SCTP to more general network settings rather than just multi-hop wireless networks. Section 3.8 concludes this chapter.

3.2 Coded CMT-SCTP

Coded CMT-SCTP is designed as an extension to the multi-path transport protocol CMT-SCTP [6]. Coded CMT-SCTP has the sender and the receiver sides. Most of the changes to the standard SCTP-SCTP scheme happens at the sender side. Coded CMT-SCTP at the sender side consists of three parts: the CMT-SCTP module, a coding module, and a group of learning agents, one for each sub-flow within the CMT-SCTP association. The coding module manages the network coding process in coordination with the learning agents on each sub-flow. The learning agent is designed to monitor and track the dynamics of the path on each sub-flow to ensure the recovery of the coded packets at the receiver side.

One of the main challenges of coding module in managing the coding process over an unreliable environment such as multi-hop wireless setting is the recovery of the original packets at the receiver despite the random losses of the network. Using *redundant packets* to cover the random losses of network and to assure the recovery of all the packets involved in the coding at receiver side is one of the known techniques in network coding. However, one of the main issues of the available network coding proposals is that they simply assume *a prior* knowledge of the path loss rate and inject redundant packets based on the assumed loss rate [86][87]. Our proposal addresses this issue by using a Q-learning agent to estimate the number of redundant packets needed to mask the random losses of the environment. We go over the details of coded CMT-SCTP at both sender and receiver side in Sections 3.2.1 and 3.2.2, respectively.

3.2.1 Sender side

The sender side of coded CMT-SCTP is composed of the standard CMT-SCTP suite, coding module and a group of Q-learning agents (one per sub-flow of the association).

Coding module

The coding module is responsible for generating the coded packets from the original packets and to keep track of original packets within the sender buffer. The coding module creates a coded packet p_k in the form of $p_k = \Sigma_i \alpha_i p_i$ where α_i is a randomly picked coefficient and p_i is the i^{th} original packet in the send buffer. The coded packet can be either redundant or innovative. The *redundant packets* carry linear equations with the same unknown set as the previous coded packet but with different random coefficients. When a coded packet is not redundant, i.e., it carries a different set of unknowns compared to the previous coded packet, it is called an *innovative packet*. The details of our coding algorithm at the sender side is presented in Algorithm 3.1. In Algorithm 3.1, destination$_i$

Algorithm 3.1 Coding Module at Sender Side

1: $numSacked = 0$
2: **for** each sub-flow$_i$ **do**
3: $numSacked_i = 0, numInnovative_i = 0$
4: $numRedundant_i = 0, numSent_i = 0$
5: **end for**
6: **if** packet from application layer **then**
7: **if** $destination_i$ is the next destination **then**
8: **while** $CWND_i$ has room **do**
9: **if** $rBuff$ has room **then**
10: $numSent_i = numInnovative_i + numRedundant_i$
$action = Qlearn_i(numSeen, numSeen_i,$
11: $numInnovative_i, CWND_i, numRedundant_i,$
$RTT_i, numSent_i)$
12: **if** action is *send an innovative packet* **then**
13: $numInnovative_i++$
14: Create an innovative packet $p = \Sigma\alpha_i p_i$
15: **else**
16: $numRedundant_i++$
17: Create a redundant packet $p = \Sigma\alpha_i p_i$
18: **end if**
19: Add packet p to send buffer
20: Send packet p to IP layer
21: **end if**
22: **end while**
23: **end if**
24: **else if** SACK arrives from the receiver **then**
25: Update $numSacked$ as $numSeen$
26: Update each $numSacked_i$ as $numSeen_i$
27: Dequeue send buffer up to $numSacked$
28: **if** Duplicate SACK arrives **then**
29: retransmit the missing packets
30: **end if**
31: **end if**

is the destination address bound with sub-flow$_i$, $CWND_i$ is the congestion window of sub-flow$_i$, and $rBuff$ is the receive buffer size for the entire association.

Q-learning agent

Each sub-flow within the CMT-SCTP association is equipped with a stand alone Q-learning agent that monitors the path features such as RTT, number of *seen* packets so far, number of sent packets so far, and the congestion window size of the path. The learning agent considers the network as an MDP with two states: *decodable* and *non-decodable* (Figure 3.1).

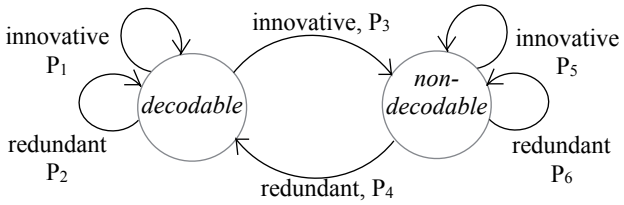


Figure 3.1: Markov Decision Process of coded CMT-SCTP. When the transition probabilities $P_1, P_2, ..., P_6$ are unknown, Q-learning is used to learn the reward on each state-action pair.

In the decodable state, there are enough coded packets (independent linear equations) for the receiver to recover the original packets and the receiver is able to decode the coded packets. In the non-decodable state, due to random losses of the medium, there are not enough coded packets (independent linear equations) for the receiver to recover the original packets. Obviously, the proposed model is an MDP because *(i)* the transition probabilities only depend on the previous state of the system and *(ii)* the Markovain rule holds for the above model. We use learning agents in our coding algorithm to control the number of redundant packets per sub-flow and to make sure that the receiver is able to decode the packets and recover all the original packets sent. Moreover, the Q-learning agent uses a classifier to determine the state of the system as either *decodable* or *non-decodable*. Each

Q-learning agent uses logistic regression classifier [99] to determine the state of the system. Then, the state of the system along with the Q-learning rule in equation (2.1) determines the action to be taken. The Q-learning take actions by informing the coding module to send either a *redundant* packet or an *innovative* one.

When the sender plans to transmit a packet, it makes an inquiry to the Q-learning agent to determine whether a redundant or innovative packet has to be sent. Upon arrival of an inquiry, the Q-learning agent determines the state of the network based on the *observed variables* from the network using the logistic regression classifier. The set of observed variables include the number of seen packets, number of sent packets so far, round trip time, congestion window size, number of innovative and redundant packets per sub-flow and number of seen packets for the whole association. The observed variables are accessible by SCTP and the learning agent to classify the network into either decodable or non-decodable state. The learning agent determines the next action (that is creating an innovative packet or redundant packet (Figure 3.1) based on the Q-learning rule of equation (2.1) and informs the coding module. Accordingly, the coding module creates a (redundant or innovative) coded packet by adding coding coefficients to the original packets involved in the coding and then sends the coded packet to the IP layer. To guarantee the orderly arrival of the packets at the receiver, we buffer the packets at the sender in First-In First-Out (FIFO) fashion prior to the coding process. When the sender plans to transmit another packet, the Q-learning agent determines the new state of the system and calculates the reward of the last action and updates the Q-matrix.

We define the reward function as a Gaussian with the parameter d_s in such a way to follow the difference between the number of innovative packets and number of seen packets in each state and reward/penalize the agent for any changes in the gap size as in equation (3.1). This way, the agent gets rewarded or penalized as the difference in equation (3.1)

gets smaller or bigger, respectively.

$$d_s = number\ of\ innovative\ packets - number\ of\ seen\ packets \tag{3.1}$$

Since the learning agent is located at the sender module in the transport layer, it can stay alive even after the SCTP association is closed. The Q-matrix of the learning agent works as a form of memory; therefore, if a new association starts, the memory of the learning agent already estimates the number of redundant packets precisely. Algorithm 3.2 is for the learning agent process. In Algorithm 3.2, in line 11, s is the current state, a is the action taken in state s, s' is the state that the agent ends up after taking action a'. $Q_t(s', a')$ is the speculation on future rewards. α and γ are the learning rate and discount factor, respectively. We choose $\alpha = \frac{1}{(1+t)^{0.77}}$ to increase the convergence rate to a satisfactory level according to [90]. We choose $\gamma = 0.9$ according to [90] [89] to stay near the optimal zone while encouraging the agent to explore more states.

In Algorithm 3.2, the number of redundant or innovative packets is not determined by the actions; the action only tells the CMT-SCTP to either "send a redundant packet" or "send an innovative packet". However, the classifier uses the number of redundant or innovative packets so far as a feature to determine the decodability or non-decodability state of the system. Hence, the classifier does not change the definition or feature of any state. As an example, an agent might be used to guide a robot to a goal state using right or left commands as actions. Then, the MDP might use the number of right and left commands so far as a feature to determine if the robot is moving towards the right state.

3.2.2 Receiver side

Coded CMT-SCTP at the receiver side consists of a decoding module and a slightly changed CMT-SCTP.

Algorithm 3.2 Learning Agent of sub-flow$_i$

{Initialization}
1: $\gamma = 0.9$, $\alpha = \frac{1}{(1+t)^{0.77}}$
2: Set matrix Q_i to zero
{Execution of the Learning Agent module}
3: **if** first decision epoch **then**
4: Determine the state s, using logistic regression classifier [99]
5: Choose an action that maximizes $Q_{t+1}(s', a')$
6: Notify CMT-SCTP on new action, i.e., redundant or innovative
7: **end if**
8: **for** each decision epoch except the first one **do**
9: Determine the state s, using logistic regression classifier [99]
10: Calculate the reward based on s, and the action a
11: Update Q-learning rule [89]

$$Q_{t+1}(s,a) = (1-\alpha)Q_t(s,a) + \alpha[r(s,a) + \gamma \max_{a'} Q_t(s',a')] \tag{3.2}$$

12: Calculate all possible $Q_{t+1}(s', a')$ where s' is the next possible state and a' is the set of available actions in that state.
13: Choose an action that maximizes $Q_{t+1}(s', a')$
14: Notify CMT-SCTP on new action, i.e., redundant or innovative
15: **end for**

Decoding module

At the receiver side, upon arrival of a new packet from any of the paths, if the new linear equation carried by the packet is independent of the existing equations (that is if an innovative packet arrives) and results in "see"-ing the next packet, the packet is buffered and the coding coefficients are retrieved from the packet header and added to the decoding matrix as a new row. The term "seen" was first introduced by Sundararajan *et al.* [87]. Based on Sundararajan's definition, packet p_i is seen when the receiver admits a packet with a linear equation in the form of $p_i + \Sigma_j \alpha_j p_j$ in which $j > i$. After adding each new row, a Jordan-Gaussian elimination process is deployed to the decoding matrix to retrieve any original packet and deliver the decoded packet to the upper layer. The decoding module, then, informs the CMT-SCTP protocol suite on the number of seen packets so far. Algorithm 3.3 summarizes the decoding module mechanism on the receiver side.

Algorithm 3.3 Receiver Side

```
    {Initialization}
 1: numSeen = 0
 2: for each sub-flow_i do
 3:    numSeen_i=0
 4: end for
    {Handling Events}
 5: if data packet from sender then
 6:    if Innovative packet arrives then
 7:       numSeen++
 8:       numSeen_i++
          {where i is the sub-flow the innovative packets arrives from}
 9:       Add the packet to the receive buffer
10:       Retrieve the coding header
11:       Add the linear combination to the decoding matrix and perform Gauss-Jordan elimination
12:       Pass any newly decoded packet to the upper layer
13:       Inform CMT-SCTP suite of numSeen and each numSeen_i
14:    else if Redundant packet arrives then
15:       Drop the packet
16:    end if
17: end if
```

Changes to standard CMT-SCTP

Coded CMT-SCTP does not change the selective acknowledgment mechanism of standard CMT-SCTP. However, when the receiver sends a SACK, instead of inserting the expected TSN number, the receiver inserts the expected number of seen packets in the acknowledgment field (based on Sundarajan's degree of freedom concept) [86][87].

3.3 A Queuing Theory Based Proof on the Performance Gain of Coded CMT-SCTP

We use a queuing theory approach to prove that network coding can effectively solve the receive buffer blocking issue. To provide a fair comparison, we use a concept called the *virtual queue* at the receiver side and compare the average number of packets in the virtual

queue in both standard CMT-SCTP and the coded CMT-SCTP. In network coding, the virtual queue is defined as the difference between the number of original packets involved in the coding process and the number of seen packets so far. Virtual queue shows how many packets are required to decode a group of original packets. Decoding at the receiver happens every time the length of the virtual queue becomes zero. Then, the receiver empties its receive buffer by delivering all the decoded packets to the application layer (called *unloading of receive buffer*). Therefore, the average length of the virtual queue in a coded multi-path transport layer is a reasonable metric to monitor the unloading of receive buffer. We adapted the virtual queue concept to the standard CMT-SCTP and coded CMT-SCTP (called it *re-sequencing queue*) to make the comparison possible. We defined the re-sequencing queue as the difference between the string sent by the sender side and the one received at the receiver side. Imagine that sender has the following packets in the buffer with TSNs of 1,2,3,4,5,6. The sender transmits the packets and the receiver gets the following string 1,2,4,5,6. Although the receiver has all the packets except 3, the transport layer can only deliver packets 1 and 2 to the application layer and packets 4,5,6 have to stay in the receive buffer. Comparing the sender string and the receiver string, there is a difference of 1 between the two strings (packet 3 is missing) and the difference causes the receiver to stall other packets in the buffer. When the difference between the sender string and receiver string is 0, i.e., there is no difference between the receiver and sender string, then the receive buffer gets emptied.

3.3.1 Virtual queue in the coded multi-path transport layer

We use the findings of [100] to calculate the virtual queue characteristics. We assume that packets arrive to the sender with a Bernoulli process at rate λ. This is a safe assumption considering the fact that most of network queuing theory modeling assumes a Poisson arrival process which is the continuous form of Bernoulli distribution. In concurrent multi-

path data transfer, we have multiple transmission paths between the sender and receiver; if we assume the i^{th} transmission path delivers the data with probability of μ_i in each time slot (exponential distribution by definition), then all transmission paths hold an exponential distribution with mean service rate of $\mu = \sum \mu_i$ in coded CMT-SCTP. The Markov chain for the length of the virtual queue is presented in Figure 3.2.

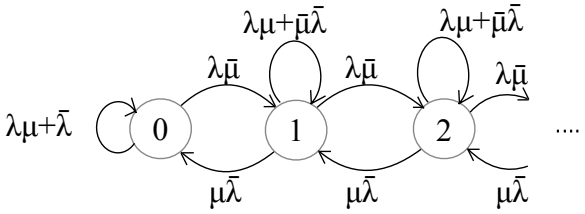


Figure 3.2: Markov chain for the length of virtual queue in coded CMT-SCTP.

If $\rho = \frac{\lambda}{\mu} < 1$, then the chain is positive recurrent and the steady state probability of the virtual queue is $\pi_i = (1-\alpha)\alpha^i$ [101], where $\alpha = \frac{\lambda(1-\mu)}{\mu(1-\lambda)}$. Consequently, the steady state expected length of the virtual queue ($E\{q_v(t)\}$) is calculated as in equation (3.3):

$$E\{q_v(t)\} = \sum_{i=0}^{\infty} i.\pi_i = (1-\mu)\frac{\rho}{1-\rho} \tag{3.3}$$

Then, the average time it takes for the receive buffer to unload is proportional to the steady state expected length of the virtual queue in equation (3.3).

3.3.2 Re-sequencing queue in standard multi-path transport layer

Based on [102], the orderly transmission of the packets over a channel can be depicted as a tandem network as in Figure 3.3. The first server represents the channel behavior and the second server represents the re-sequencing behavior of the receive buffer. The first queue associated with the first server is basically representing all the different paths between the sender and the receiver. Because we assume an exponential distribution for the i^{th}

transmission path erasure behavior, multiple queues can be substituted with one queue with an exponential behavior. We know the arrival rate of the first queue, but not the arrival rate of the second queue which is equal to the departure rate of the first queue. The Burke's theorem [102] offers a great solution for sequential queues. According to Burke's theorem, for a $M/M/1$ queue, $M/M/c$ or $M/M/\infty$ queue in a steady state, the arrival rate is the same as the departure rate [102]. Using Burke's theorem [102], the arrival rate of the second queue is λ; that is, same as the departure rate of the first queue and is independent of the service rate of the first queue. As such, we can analyze the behavior of both queues separately.

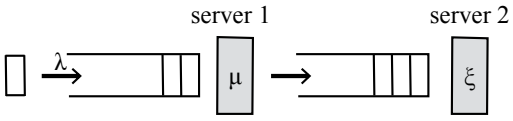


Figure 3.3: Queuing schematic for the re-sequencing process in standard multi-path data transfer; server 1 models the channel behavior while server 2 models the re-sequencing behavior of receive buffer.

The Markov chain of the first queue in Figure 3.3 is exactly the same as that in Figure 3.2. If we assume that the service rate of the second queue is ξ, i.e., in-order packet arrivals have the probability of ξ and out of order packet arrivals have the probability of $1-\xi$, then the Markov chain for the re-sequencing behavior of the receive buffer (i.e., the second queue in Figure 3.3) looks like that in Figure 3.4.

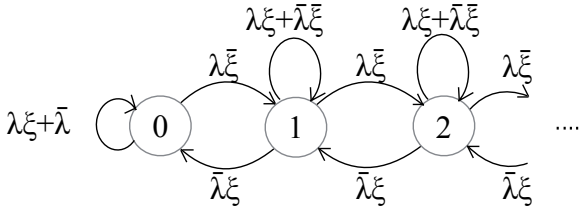


Figure 3.4: Markov chain for the re-sequencing behavior of the receive buffer.

If $\rho_1 = \frac{\lambda}{\mu}$ associates with the first queue and $\rho_2 = \frac{\lambda}{\xi}$ associates with the second queue,

then based on [101], the steady state probability of each queue can be calculated as equation (3.4) and (3.5), respectively.

$$\pi_i^1 = (1-\alpha_1)\alpha_1^i, \quad \alpha_1 = \tfrac{\lambda(1-\mu)}{\mu(1-\lambda)} \tag{3.4}$$

$$\pi_i^2 = (1-\alpha_2)\alpha_2^i, \quad \alpha_2 = \tfrac{\lambda(1-\xi)}{\xi(1-\lambda)} \tag{3.5}$$

Because the behavior of the two queues are independent of each other, the total number of re-sequenced packets ($E\{q_r(t)\}$) can be calculated according to equation (3.6).

$$\begin{aligned} E\{q_r(t)\} &= E\{size\ of\ queue_1\} + E\{size\ of\ queue_2\} \\ &= \sum_{i=0}^{\infty} i.\pi_i^1 + \sum_{j=0}^{\infty} j.\pi_j^2 \\ &= (1-\mu)\frac{\rho_1}{1-\rho_1} + (1-\xi)\frac{\rho_2}{1-\rho_2} \end{aligned} \tag{3.6}$$

$E\{q_r(t)\}$ shows the average amount of difference between the actual sequence transmitted by the sender and the one that arrives at the receiver. A higher $E\{q_r(t)\}$ results in higher receive buffer occupancy rate and longer waiting time for the packets to be transfered to the upper layer. Comparing equations (3.3) and (3.6), one can easily conclude $E\{q_r(t)\} > E\{q_v(t)\}$. That is, on average the receive buffer of the coded transport layer unloads more frequently and the wait times of the packets are less as compared to those of the standard CMT-SCTP. Therefore, we conclude that the probability of receive buffer blocking issue in the coded CMT-SCTP is smaller as compared to in the standard CMT-SCTP.

3.4 Testbed Experiments For Accuracy Analysis Of The Q-learning Algorithm

In our earlier work [1], we used a naive Bayes classifier with the Q-learning agent to determine the state of the network. However, the performance of the naive Bayes classifier can be compromised because of the unrealistic assumption that all features are equally important and independent given the value of the class. To increase the accuracy of each Q-learning agent in determining the state of the system, naive Bayes classifier is substituted by logistic regression [99]. Although logistic regression is not as biased as naive Bayes towards the features, it suffers from over-fitting. We propose using logistic regression classifier with some correction to the prior estimation and data sampling to prevent the over-fitting behavior. In this section, a comparison between the performance of the naive Bayes classifier and logistic regression classifier is studied via testbed experiments.

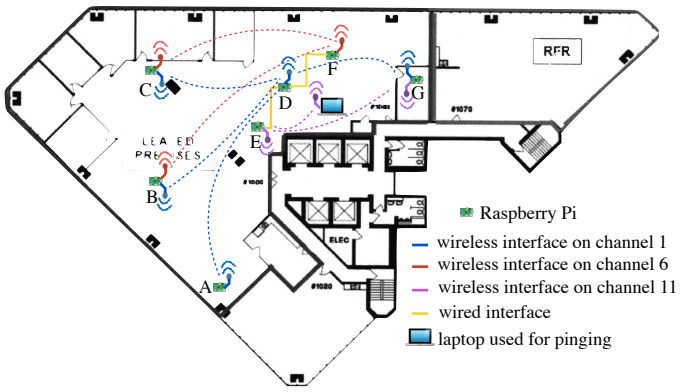


Figure 3.5: Testbed topology - showing multiple data paths for CMT-SCTP.

For evaluation purposes, we used the testbed setup described in 2.3 and Figure 3.5. In terms of the data traffic, we created two multi-homed CMT-SCTP associations between nodes B-G and C-G. In addition, we created a single-homed CMT-SCTP association between nodes A-G. Nodes A, B, and C each send 10,000 SCTP packets to node G during the experiment. Each SCTP packet carries 1000 bytes of application data. The data paths

are as follows:

- node A: A-D-G
- node B: B-D-G and B-F-D-E-G
- node C: C-D-G and C-F-D-E-G

To guarantee path discrepancies, we created two types of interference traffic in the network. One interference traffic is generated by the two CMT-SCTP associations set up between nodes A-G and C-G to create path discrepancies between the sub-flows of the CMT-SCTP associations between nodes B-G and C-G. Nodes A and C send SCTP packets carrying 1000 bytes of application data to node G during the experiment. The other interference traffic is generated by a laptop with a wireless interface on channel 11. The laptop pings the wireless interface of node E at different rates of 10, 20, 30, 40, 50 ping packets per second. Ping packets contribute as a controllable source of interference traffic to create path discrepancies for the sub-flows of the CMT-SCTP association between nodes B-G.

Because we investigated the accuracy of the classifier within the learning agent which is in charge of determining the state of the packet transmission over the path as either lost or received, we collected the training data and label them based on the receiver (node G) output on the status of each packet.

The accuracy of a machine learning algorithm is defined as in equation (3.7). However, accuracy is not the only factor in choosing the right learning algorithm for a data set. Another important factor is the sensitivity of the machine learning mechanism to the positive class incidents, especially when dealing with data sets of rare positive class. In a data set with a rare positive class population, the learning algorithm might have a high accuracy due to over-fitting and classifying all samples as negative class or the large

number of negative class samples. As such, another factor has to be measured to show the sensitivity of the learning algorithm towards the positive class incidents. The sensitivity of a learning mechanism is defined as in equation (3.8).

$$Accuracy = \frac{number\ of\ correct\ predictions}{sample\ size} \tag{3.7}$$

$$Sensitivity = \frac{correct\ predictions\ of\ the\ positive\ class}{total\ number\ of\ prediction\ of\ positive\ class} \tag{3.8}$$

Although naive Bayes holds a reasonable accuracy, the sensitivity of the learner to the the rare positive incidents is very low. Logistic regression on the other hand, when used over an endogenous data set, holds a very high accuracy and sensitivity measure. The strategy for endogenous selections is to collect the instances of the positive class and then choose randomly from within the negative class instances. Reference [99] suggests to choose as many as positive class within the samples from the negative class to create a balance training set. Logistic regression classifier with prior correction uses the pre-knowledge about the positive class to adjust its prediction. Both prior correction and endogenous sampling are methods to increase the sensitivity of the logistic regression classifier.

For our Q-learning mechanism which is responsible for the timely transmission of the redundant packets to guarantee the solvability of the coding process at the receiver side, the sensitivity of the classifier is crucial. For a coding process, not only it is important to send enough redundant packets to cover for all the random losses, but also it is crucial to send the redundant packets timely, to avoid any unwanted delay for decoding at the receiver. To show different levels of accuracy, we run naive Bayes, standard logistic regression, and logistic regression with prior correction (our proposal in this chapter) over a random sampling training set and over an endogenous training set.

All data sets collected from the testbed in Figure 3.5 are presented in Table 3.1 which

Classifier	Training set type	Accuracy	Sensitivity
Naive Bayes	random sampling	60%	1%
Naive Bayes	endogenous sampling	69%	45%
Standard logistic regression	random sampling	99%	13%
Standard logistic regression	endogenous sampling	95%	70%
Logistic regression with prior correction	random sampling	99%	13%
Logistic regression with prior correction	endogenous sampling	95%	75%

Table 3.1: Performance comparison between naive Bayes and logistic regression classifiers. summarizes the different accuracy and sensitivity levels of naive Bayes, standard logistic regression, and logistic regression with prior estimation correction. As shown in Table 3.1, logistic regression with prior correction is the logical choice that can satisfy both sensitivity and accuracy needs. Thus, we used logistic regression with prior correction in Q-learning agents' classifiers for the performance studies in Section 3.5 and 3.6.

3.5 Simulations For The Performance Of Coded CMT-SCTP

We used QualNet 7.1 [103] for our simulations. The standard CMT-SCTP module in QualNet was developed by Ilknar Aydin [74]. We used this CMT-SCTP module to develop our coded CMT-SCTP module in QualNet as well.

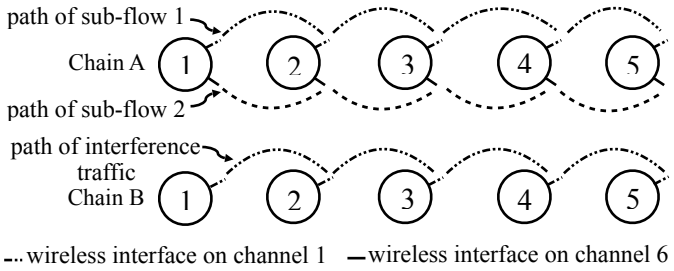


Figure 3.6: Simulation topology - Actual data flow is over chain A, while interference traffic is over chain B.

We set up a multi-hop wireless network of 5 multi-homed nodes located in a chain topology (Figure 3.6, Chain A). Each multi-homed node in the chain is equipped with two IEEE 802.11b wireless network interfaces tuned to two different channels (1 and 6) to avoid any interference among the CMT-SCTP sub-flows. The data rate for IEEE 802.11b is 2 Mbps and the RTS/CTS mechanism is on. The transmission range is around 370 meters, the interference range is around 812 meters, and the carrier sensing range is around 520 meters for both of the wireless interfaces. Neighboring nodes in the chain are 300 meters apart. A data source over CMT-SCTP is located at node 1 of chain A and continuously transmits data to node 5 of chain A. CMT-SCTP uses RTX-CWND as the re-transmission policy. Each SCTP data chunk carries 1000 bytes of application data.

As we are interested in the performance of coded vs. standard CMT-SCTP in the presence of severe path discrepancies between the CMT-SCTP sub-flows, we set up a second chain of nodes in the network (Figure 3.6, Chain B). The number of nodes in chain B is the same as the number of nodes in chain A. However, each node in chain B has only one IEEE 802.11b wireless interface operating on channel 1 and with the same wireless properties as the first wireless interface of the nodes in chain A. Chain B is located 450 meters away from the chain A. To create background traffic (i.e., interference and hence loss) for the CMT-SCTP sub-flow running on path 1 of chain A, we set up CBR (Constant Bit Rate) traffic on chain B for the entire simulation time. Each CBR packet carries 1000 bytes of data. We vary the CBR traffic rate as 0, 4, 8, 16, 24, 50, 60, 70, and 80 packets per second to create different levels of background traffic in path 1 of chain A.

Each simulation configuration is run 30 times and average values are calculated with 95% confidence intervals. The simulation time is 7 minutes and the measurements are done from 2nd to 6th minute to ensure stable results. To show that coded CMT-SCTP effectively handles receive buffer limitations, we tested our scheme with different receive buffer sizes.

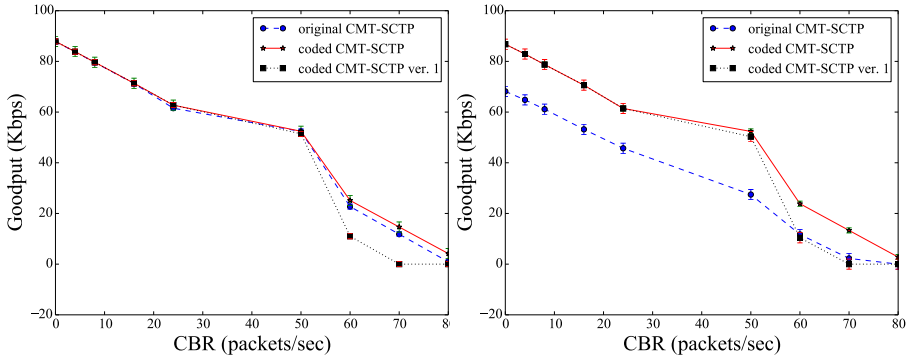
Our goal is to show that receive buffer limitation does not have a substantial effect on the performance of coded CMT-SCTP unlike the standard CMT-SCTP. Based on the findings of Aydin *et al.* [71], a small receive buffer size (less than 128KB) causes performance degradation in standard CMT-SCTP. Therefore, we ran simulations for unlimited buffer size as well as 16 KB and 8 KB receive buffer sizes.

Note that, our simulation results over the topology of Figure 3.6 can be generalized to real life multi-hop wireless networks. In real life scenarios, different number of hops or more complicated network topologies would only contribute to the path discrepancies of the sub-flows of CMT-SCTP; therefore, the simulation results based on the simulation set up we used provides a strong evidence for the validity of our coding scheme over more generalized scenarios.

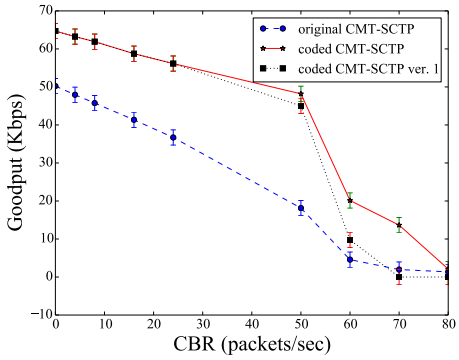
We measured the number of bytes delivered to the receiving application over time (*goodput*) of standard CMT-SCTP vs. coded CMT-SCTP to show the effectiveness of our network coding algorithms in a multi-hop wireless setting.

Figures 3.7(a), 3.7(b), and 3.7(c) compare the performance of coded CMT-SCTP vs. standard CMT-SCTP when the receive buffer size is unlimited, $rBuf = 16K$, and $rBuf = 8K$. The rule of thumb to choose receiver buffer size is that the size has to be at least the bandwidth-delay product of the path (Round trip time multiply by the link capacity). For our simulation setup, the link has a data rate of $2Mbps$; with a RTT of 10 to $20ms$, the bandwidth-delay product of the network is roughly around $2.5K$ to $5K$ bytes. Thus the receive buffer sizes of $8K$ or $16K$ are large enough, yet under $128K$ to see the receive buffer blocking issue.

As depicted in Figure 3.7, both coded CMT-SCTP and standard CMT-SCTP perform similarly with an unlimited receive buffer. However, when the receive buffer size is decreased to 16KB, the performance of standard CMT-SCTP degrades drastically while the



(a) $rBuf = maximum\ in\ Linux$ (b) $rBuf = 16K$



(c) $rBuf = 8K$

Figure 3.7: Goodput of standard CMT-SCTP vs. coded CMT-SCTP (proposed in this chapter) vs. coded CMT-SCTP ver. 1 [1] with different receive buffer (rBuf) sizes, in simulation.

coded CMT-SCTP performs as if the receive buffer size is unlimited. Our results show that coded CMT-SCTP outperforms the standard CMT-SCTP by a factor of 20% to 47%. When the receive buffer size is decreased to 8KB, the performances of both standard and coded CMT-SCTP suffer; however, coded CMT-SCTP still out performs the standard CMT-SCTP by a factor of 22% to 62%. Note that, the performance of coded CMT-CMT stays close to the ideal performance (when the buffer size is unlimited) as the receive buffer size gets smaller, showing the effectiveness of coded CMT-SCTP to solve the receive buffer blocking problem in the standard CMT-SCTP.

We also measured the performance of (the older) version 1 of our coded CMT-SCTP proposal in [1] for comparison and included the results in Figure 8. Older coded CMT-

SCTP uses a single Q-learning agent for the entire CMT-SCTP association while our current coded CMT-SCTP proposal in this chapter has a group of learning agents (one agent per sub-flow of the CMT-SCTP association). Expectedly, the performance of older and those of current versions of coded CMT-SCTP are similar when the sub-flow paths have similar loss rates (both paths have low interference). However as the discrepancy in the loss rates of the two sub-flow paths increases, the performance of older version of coded CMT-SCTP gets worse. In the presence of severe path discrepancies, the older version of coded CMT-SCTP association stalls because of over-estimating the required number of coded packets and hence flooding the network with redundant packets. However, the current coded CMT-SCTP proposal estimates the number of coded packets independently and more correctly for each sub-flow path and does not send as much unnecessary redundant packets to the network.

3.6 Testbed Experiments For Performance Of Coded CMT-SCTP

In this section we investigate the performance of our coded CMT-SCTP using the wireless testbed in Figure 3.5. The experiment configuration and data and interference traffic set up are all the same as described in Section 3.4. The objective of the experiments is to compare the performance of our coded CMT-SCTP with the standard CMT-SCTP and CMT-SCTP with buffer splitting [2] scheme.

Our main claim is that coded CMT-SCTP enhances the performance by eliminating the receive buffer blocking in presence of path discrepancies. As such, a testbed with dissimilar independent paths for each sub-flows within an CMT-SCTP association is a prefect set up for proof of concept. We measured the space left in the receive buffer of node G which is data receiver of SCTP association between nodes B and G. Each experiment is repeated 10

times with the same set of parameters to make sure that the results are accurate. Figure 3.8 shows the percentage of packets that experience receive buffer blocking during the data transmission versus the ping message rate to node E. Note that the ping messages are used to create interference and hence loss in one of the sub-flows of the B-G association.

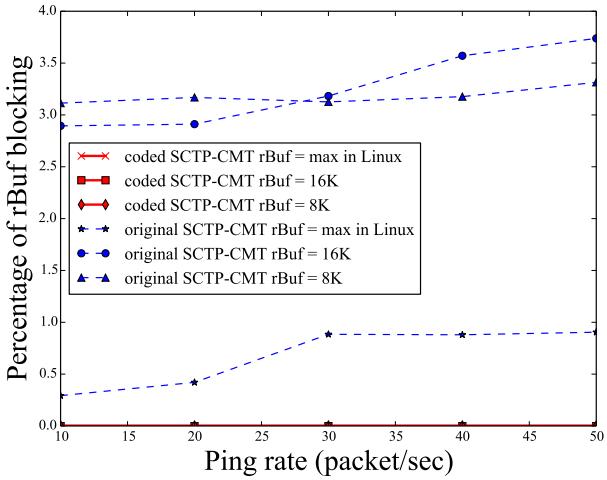
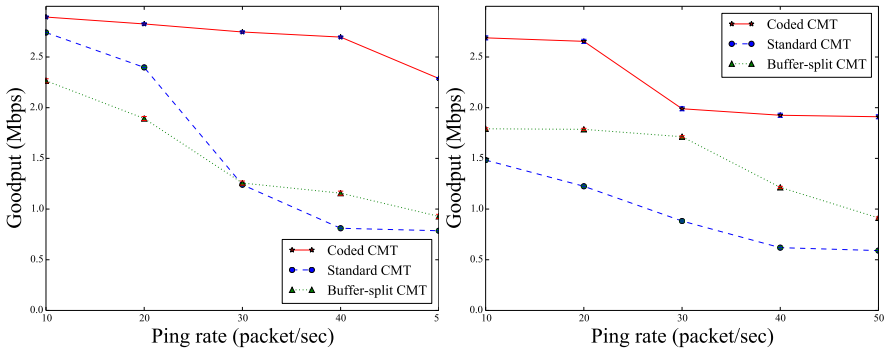


Figure 3.8: Percentage of packets in CMT-SCTP association that experience receive buffer blocking for standard CMT-SCTP vs. coded CMT-SCTP.

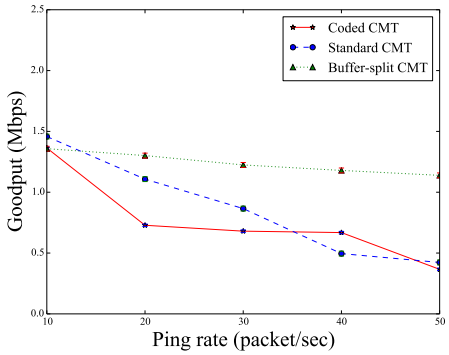
As depicted in Figure 3.8, as the receive buffer (rBuf) size decreases, the percentage of packets experiencing rBuf blocking increases for the standard CMT-SCTP. Note that, the percentage of the rBuf blocking incidents increase by 6 times for rBuf size of 8 KB as compared to the unlimited rBuf size for the standard CMT-SCTP. On the other hand, coded CMT-SCTP packets experienced no rBuf blocking during the experiment. (See the coded CMT-SCTP curves laying at zero on the x-axis in Figure 3.8)

We also measured the *goodput* (amount of bytes delivered to the receiving application over the experiment time) of the CMT-SCTP associations (see Figure 3.9).

Coded CMT-SCTP effectively increases the goodput by a factor of 50% as compared to the standard CMT-SCTP and CMT-SCTP with buffer split [2] when the receive buffer size is set to max. size in Linux. The goodput of coded CMT-SCTP becomes 40% better



(a) $rBuf = maximum\ in\ Linux$ (b) $rBuf = 16K$



(c) $rBuf = 8K$

Figure 3.9: Goodput of the standard CMT-SCTP vs. CMT-SCTP with buffer splitting [2] vs. coded CMT-SCTP with different rBuf sizes, over the testbed.

than the standard CMT-SCTP and CMT-SCTP with buffer splitting when the buffer size gets reduced to 16 KB. These results show the effectiveness of coded CMT-SCTP. However, when the receive buffer size becomes 8 KB, coded CMT-SCTP seems to lose its effectiveness as compared to the other two schemes. The reason is that the receive buffer eventually becomes too small to handle overhead caused by the redundant packets. When using CMT-SCTP, the out of order packet arrivals, not only triggers the receive buffer blocking, but also causes unnecessary retransmission of packets. The CMT-SCTP with buffer splitting seems to handle the packet retransmissions better for really small receive buffer sizes. Comparing Figure 3.9(c) with its simulation counterpart, coded CMT-SCTP seems to perform worse than the standard CMT-SCTP in the testbed experiment while it performs better than the standard CMT-SCTP in simulation results for rBuf size of

8 KB. The reason behind this behavior in the testbed is the CPU cycle of Raspberry Pi. Raspberry Pi has a "*limited CPU cycle*" [104]; as such in computationally demanding tasks, the performance of the buffer becomes a bottleneck on the overall performance as buffer size gets smaller [104]. According to Raspberry Pi Foundation, the overall performance of the Pi's processor is comparable with a PC using an Intel Pentium 2 processor clocked at 300MHz which introduces some limitation for computationally demanding tasks.

3.7 Discussion

In this section, we discuss the delay overhead of coded CMT-SCTP and also how it can be integrated into more general networking scenarios, particularly to the heterogeneous networks. Network coding increases the complexity of the mechanism by combining the packets instead of simply forwarding the packets. The decoding delay at the receiver depends greatly on the coding method at the sender. Because we used random linear network coding, the decoding delay is in the order of $O(\frac{1}{1-\rho})$ where $\rho = \frac{\lambda}{\mu}$ is the network load. λ is the arrival rate for the traffic and μ is the service rate of the channel [105–107].

In terms of integrating coded CMT-SCTP into the legacy networks, a transport protocol's end points negotiate the options of the connection during handshake and can load the necessary end-to-end modules to support wireless links. Similarly, coded CMT-SCTP option can be negotiated during handshake to work in harmony with the other legacy protocols in the network. In terms of the use of coded CMT-SCTP in heterogeneous networks, there are two main approaches in designing a transport protocol that supports wireless links in the network. The first approach is a transport connection which is established over the end-to-end path that may have wireless links [108]. The second approach is to deploy a gateway to separate the wireless and wired portions of the network [109]. Our coded CMT-SCTP can be adapted to work with both of these approaches. For the first

approach, the proof of concept is already provided in this chapter with the experiments using the heterogeneous testbed in Figure 3.5. However, more in-depth investigation is needed to fine-tune the Q-learning algorithm to choose features that fully represent the mixed network characteristics. The second approach is more practical, as the wireless part of the network can benefit from the better performance of the network while it is shielded from the wired part of the network. The proxy gateway basically creates a shield between the two parts and provides the necessary changes from one protocol to another when the packet passes through the gateway.

3.8 Conclusion

Network coding is shown to enhance the capacity of multi-hop wireless networks. In this work, we showed that network coding can also be used as a tool to eliminate the inherent receive buffer blocking issue during the concurrent multi-path transfer of data and hence improve throughput. We proposed an adaptive network coding scheme for CMT-SCTP, called coded CMT-SCTP, and used an adaptive Q-learning algorithm to control the number of redundant packets to effectively recover the lost packets. Our coded CMT-SCTP is highly effective in alleviating the receive buffer blocking. Our coded CMT-SCTP outperforms the standard CMT-SCTP in terms of throughput up to 62% depending on the receive buffer size and path dissimilarities. Testbed findings support the simulation and theoretical results.

Chapter 4

A Smart Fairness Mechanism for Concurrent Multipath Transfer in SCTP over Wireless Multi-hop Networks

4.1 Introduction

In this chapter, we studied the fairness behavior of CMT-SCTP on a multi-hop wireless testbed. We introduced a Q-learning distributed mechanism to enhance fairness in SCTP association. The proposed method uses RL mechanism to acquire knowledge about the dynamics of the network. Consequently, the acquired knowledge is used to choose the best action to improve the fairness index of the network. We evaluated our proposal against the standard CMT-SCTP and resource pool CMT-SCTP (CMT/RP-SCTP). Our proposal outperforms the available fairness mechanisms for CMT-SCTP by a significant margin. Fairness of CMT-SCTP is still an open issue and well under discussion as there exists so many unanswered questions on this topic. Our investigation contributes to the field in three ways:

(a) It is a known fact that the window based transport protocols are not designed for the imperfections of the wireless multi-hop network and tend to act unfairly against flows coming from nodes farther away from the gateway [110], [48], [55]. SCTP is not an exception as well; however, when using CMT-SCTP, the goal is not to compromise fairness of the base protocol to get a higher bandwidth utilization or throughput.

As such, fairness consideration has to be integrated within any adds-on algorithm in multipath transport protocols. The above unaddressed concern motivated us to evaluate the fairness of CMT-SCTP against non-CMT flows that are coming from farther away hops in a multi-hop wireless environment. Our investigation reveals that CMT-SCTP is indeed unfair towards flows coming from farther away hops.

(b) We used our findings to develop a dynamic distributed fairness mechanism which alleviates the aggressive behavior of CMT-SCTP towards non-CMT flows coming from farther away hops.

(c) Our state of the art fairness mechanism, Q-learning CMT-SCTP, enhances the fairness behavior of CMT-SCTP tremendously. We compare the performance of our proposal with the standard CMT-SCTP and CMT/RP-SCTP; the comparison confirms that Q-learning CMT-SCTP significantly outperforms the available fairness mechanisms. Moreover, Q-learning CMT-SCTP is capable of supporting different levels of quality of service (QoS) based on user demand.

The present chapter is divided into the following parts: Section 4.2 presents the results of investigation and measurements of CMT-SCTP in different scenarios (part (a) of the contribution). In Section 4.3, the proposed dynamic distributed fairness mechanism is explained elaborately. The evaluation of the proposed algorithm and a brief discussion on comparable available mechanisms is presented in Section 4.4. Section 4.6 concludes this chapter.

4.2 Study of CMT-SCTP Fairness

To investigate the fairness of CMT-SCTP and compare it with SCTP, we set up a multi-hop wireless testbed and measured the average throughput of flows in different scenarios

(refer to Section 2.3 for testbed setup details). To create a test-bench for comparison, the average throughput of node B is measured when sending data over SCTP. Then, the average throughput of node B is measured when sending data over CMT-SCTP. Comparing the two values gives us a good insight on how aggressive CMT-SCTP is compared to its based protocol SCTP.

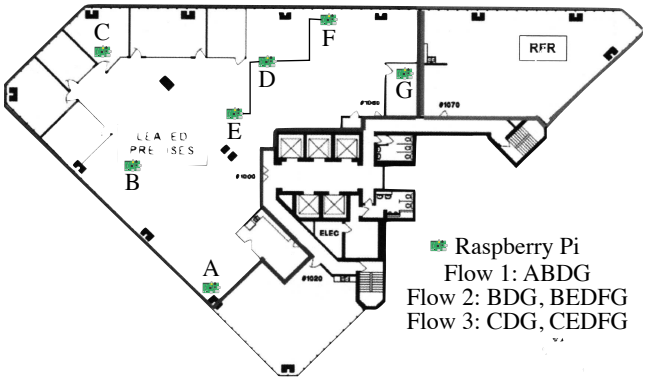


Figure 4.1: Testbed topology.

Figure 4.1 shows the testbed topology. Node A has only one wireless interface; while each of nodes B, C, and G has two wireless interfaces. Node D is a forwarder node that has one wireless interface. Each of nodes E and F has one wireless interface. The testbed is setup in a way that the first interface of nodes B, C and G communicates with node D. The second interface of nodes B and C communicates with node E, while the second interface of node G communicates with node F. The data collections took place in different hours of the day to make sure that all possible outcomes are covered in our data set.

We measured network statistics in 5 different scenarios:

- scenario 1: all source nodes use TCP
- scenario 2: all source nodes use TCP, node B uses single-homed SCTP
- scenario 3: all source nodes use TCP, node B uses CMT-SCTP with two interfaces

- scenario 4: all source nodes use single-homed SCTP
- scenario 5: all source nodes use single-homed SCTP, node B uses CMT-SCTP

Scenario 1 is designed to act as a benchmark to get a better insight on fairness behavior of CMT-SCTP as compared to SCTP and TCP. Scenario 2 is designed to reveal the difference between TCP and SCTP performance over a wireless multi-hop setting. Scenario 3 is designed to reveal any unfairness in CMT-SCTP in comparison to its base protocol SCTP in scenario 2. In each of these scenarios, we only changed one parameter on one node to make sure that any changes in the result is caused by the transport layer performance. Scenarios 4 and 5 are designed to monitor the fairness behavior of CMT-SCTP against single-homed SCTP. The measurement results from the above experiments provide us with enough evidence on CMT-SCTP fairness. All scenarios are run for 50 times to create an acceptable result within the 95% percentile confidence interval.

To compare performance of difference scenarios, we measured the throughput of each flow in *kbps* and Jain's fairness index in each scenario. Jain's fairness index is defined as [111]:

$$J(x_1, x_2, ..., x_n) = \frac{(\Sigma_{i=1}^{n} x_i)^2}{n\Sigma_{i=1}^{n} x_i^2} \tag{4.1}$$

where x_i is the data rate of the i_{th} flow. The measurement results of the above scenarios are presented in Table 4.1.

Table 4.1 represents the average throughput of the competing flows in the above scenarios. Looking into the first column of Table 4.1, when both nodes A and B use TCP to transfer data, node B which is located in a closer hop-count from node G, i.e. destination, has a higher throughput. We are not going to elaborate on why TCP is not fair to flows with different number of hops to destination as there exist sufficient literature on the reasons behind this behavior. The measurements in scenario 1 and scenario 2 show that TCP and single-homed SCTP have similar fairness behavior when it comes to the

source node	scenario 1	scenario 2	scenario 3	scenario 4	scenario 5
A	1050.54	822.55	225.94	396.68	120.33
B	4304.12	2892.51	3370.45	2962.86	2398.18

Table 4.1: Throughput comparison of different scenarios. In scenario 1 of Figure 4.1, nodes A and B send data using TCP. In scenario 2, all sources except node B use TCP; node B uses SCTP. Scenario 3 is similar to scenario 2; however, node B uses CMT-SCTP. In scenario 4, all nodes use SCTP. Scenario 5, all nodes use SCTP, node B uses CMT-SCTP. All the above numbers are measured in kbps.

flows coming from farther away nodes. SCTP and TCP share similar congestion control mechanism, as such similar results are expected in scenario 1 and 2. However, as stated in Table 4.1, SCTP on node B has lower throughput as compared to TCP in scenario 1. The reason behind the drop in throughput of SCTP is the processing power of Raspberry Pi. SCTP uses more CPU cycle per transfer as compared to TCP [112]; therefore, the SCTP throughput takes a plunge in scenario 2. Our measurements showed that when there are no other flows in the testbed, TCP source on node A can have an average throughput of 1443 *kbps*, which is 30% more than the throughput of node A in scenario 1.

Measurements of scenario 2 show that there is not a huge difference between performance of single-homed SCTP and TCP in terms of fairness towards flows coming from nodes farther away from destination. The measurement results in Table 4.1 confirms that CMT-SCTP (scenarios 3 and 5) starves flow A more as compared to single-homed SCTP (scenarios 2 and 4). The third column in Table 4.1 shows that the average throughput of CMT-SCTP in node B is 16% higher than the average throughput of SCTP in node B in scenario 2. Having more than one interface provides node B with more opportunities and eventually a higher share of network resources, which causes an 80% drop in average throughput of node A as compared to scenario 1, and a 70% drop when compared to scenario 2. It is fair to conclude that CMT-SCTP has a more aggressive congestion control

mechanism as compared to SCTP when it interacts with non-CMT flows. In scenario 4, all source nodes use single-homed SCTP to send data. Measurement results depicted in Table 4.1 for scenario 4 show that the SCTP congestion control mechanism is more aggressive towards other SCTP flows that are coming from farther away hops as compared to TCP. Scenario 5 shows that CMT-SCTP is even more aggressive towards other non-CMT flows as compared to SCTP and TCP.

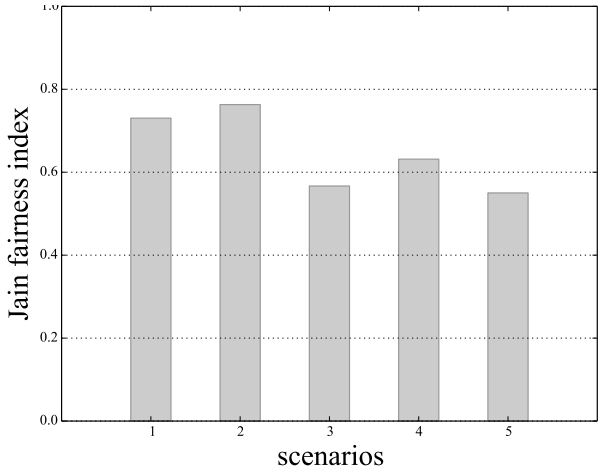


Figure 4.2: Fairness comparison of different scenarios. In scenario 1, node A and B in Figure 4.1 send data using TCP. In scenario 2, all sources except node B use TCP; node B uses SCTP. Scenario 3 is similar to scenario 2; however, node B uses CMT-SCTP. In scenario 4, all nodes use SCTP. Scenario 5 is similar to 4 except for node B that uses CMT-SCTP.

Figure 4.2 depicts Jain's fairness index for different scenarios; the Jain's fairness index has not changed drastically from scenario 1 to scenario 2. Scenario 3 reveals the impact of CMT-SCTP in comparison with SCTP.

To take a closer look into the factors that contribute to the unfair behavior of CMT-SCTP, we monitor some of the parameters of the transport layer and realize that the size of congestion control window plays a crucial role in the outcome, as shown in Figure 4.3. The congestion window size of the SCTP association on node B in scenarios 2, 3, 4, and 5 is monitored in all experiments via sampling. The average congestion window size is then

calculated for the experiments in scenarios 2, 3, 4, and 5 and graphed for each experiment (we only include 40 experiments on the x axis). The solid lines in Figure 4.3 show the average congestion window size of node B when using CMT-SCTP in scenarios 3 and 5, while the dashed lines represent the average congestion window size of node B when using SCTP in scenarios 2 and 4. The difference between the solid and dashed lines shows the aggressive behavior of CMT-SCTP as compared to SCTP in presence of non-CMT flows.

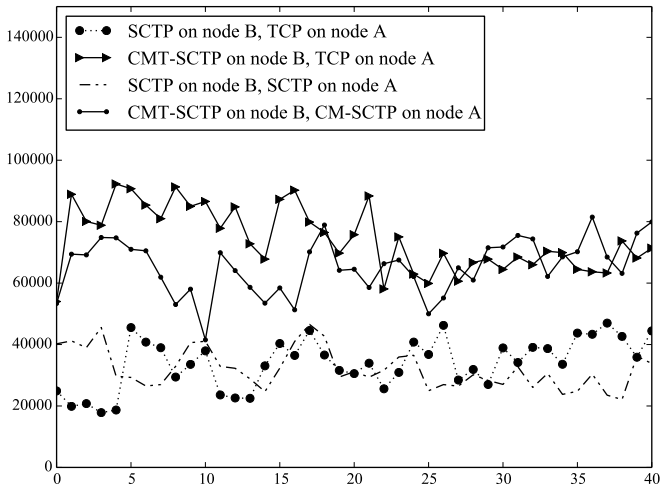


Figure 4.3: Comparison of the average congestion window size of node B in scenario 2 and 3 of Figure 4.1. The solid line shows the average window size of node B when all sources are using TCP and node B used CMT-SCTP; while the dashed line shows the average window size of node B when all source nodes are using TCP and node B uses SCTP.

It is expected that utilizing multiple paths in CMT-SCTP provides the user with higher throughput; however, the aggregate throughput on all interfaces has to be comparable with a single-homed SCTP to maintain fairness among the flows. Based on our findings in this section, CMT-SCTP aggressive congestion control window behavior in maximizing the aggregate bandwidth causes other non-CMT flows coming from farther away hops to suffer and struggle for bandwidth. The results of our investigation motivated us to propose a dynamic mechanism to control the maximum congestion window size as an effective tool

to alleviate the aggressive behavior of CMT-SCTP against other transport layer protocols.

4.3 Q-learning CMT-SCTP

The investigation in Section 4.2 shows that CMT-SCTP is unfair to non-CMT flows in a multi-hop wireless environment, especially to those farther away from the destination. Considering that CMT-SCTP on each node deals with an unpredictable environment over the wireless multi-hop network, a dynamic fairness solution is required to address the unfair behavior of CMT-SCTP. Reinforcement learning methods are the best candidate for an interactive solution capable of adjusting with the changes of the environment as the system runs. Among various reinforcement learning methods, Q-learning fits our needs the best, as it is a model-free technique that can be used to find an optimal action-selection policy for any given finite state MDP.

In this section, we go over our dynamic distributed fairness mechanism, Q-learning CMT-SCTP, which uses machine learning and network statistics to determine fairness of CMT-SCTP. Our distributed fairness mechanism controls the aggressive behavior of CMT-SCTP by using a dynamic damp on the SCTP maximum congestion window.

4.3.1 Features and states

At the beginning of each decision epoch, the learning agent in the CMT-SCTP source node receives transport layer statistics including round trip time (RTT_i), congestion control window size ($cwnd_i$), retransmission timeout (RTO_i), and flight size (FL_i) on all interfaces. RTT_i, RTO_i, $cwnd_i$, and FL_i are the statistics of the i_{th} interface that are fed to the learning agent. Subsequently, the agent uses a Bayes classifier to determine the state of the network as the source node sees it. The MDP inside each CMT-SCTP source node has the following four states, as shown in Figure 4.4:

- state 1: the network condition is classified as (fair,unfair) in decision epoch $(t-1,t)$.
- state 2: the network condition is classified as (unfair,fair) in decision epoch $(t-1,t)$.
- state 3: the network condition is classified as (unfair,unfair) in decision epoch $(t-1,t)$.
- state 4: the network condition is classified as (fair,fair) in decision epoch $(t-1,t)$.

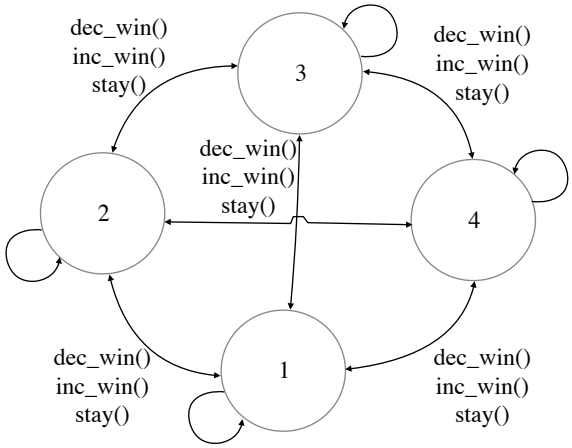


Figure 4.4: The MDP of CMT-SCTP Q-learning agent. The learning agent can choose one of the actions in each state based on the Q-learning rule.

At the beginning of the learning process, the learning agent does not know the transition probabilities of the MDP and the associated reward with each transition. To find the transition probabilities of the MDP, we use the Q-learning algorithm [88].

4.3.2 Actions

After the agent determines the state of the network in each decision epoch, the CMT-SCTP learning agent chooses an action from the action set $A = \{decWin(), incWin(), stay()\}$ and informs CMT-SCTP to adjust the maximum congestion window size. The functions in the action set A are the tools for communicating the learning agent policies to the

transport layer. $decWin()$ decreases the maximum congestion window size of the CMT-SCTP association to half of its current size. $incWin()$ increases the maximum congestion window size of the CMT-SCTP association to 1.5 times of its current size. $stay()$ maintains the maximum congestion window size at its current value.

4.3.3 Reward function

The learning agent uses a reward function to receive feedback on the consequences of taking an action. In the next decision epoch, the environment responds with the new state. Based on the new state of the MDP, the learning agent inside CMT-SCTP receives a reward. The reward function is in form of Eq. (4.2):

$$r_t = e^{\frac{tp_t - T}{\sigma}} \tag{4.2}$$

where tp_t is the throughput of the CMT-SCTP association during decision epoch t. T is the estimated threshold for fair bandwidth share of CMT-SCTP association.

The reward function plays an important role in the convergence rate of the Q-learning, and studies suggest that Gaussian reward functions can effectively increase the convergence rate of the learning while directing the agent toward the goal state [91]. As such, we choose the reward function in the form of equation (2). Moreover, the chosen action set in subsection 4.3 provides the agent with the enough tools to control the transport layer dynamically based on the characteristics of the network. Choosing the right action set is very critical in the design of the underlying MDP. Providing the Q-learning agent with a large set of actions in each state results in a prolong convergence time. As such, we chose to have three actions in each state to shorten the optimization period (increase the convergence rate) while finding the right maximum congestion window size for CMT-SCTP to act fairly toward other flows.

The learning process continues as long as the network is up and running; it basically works as a memory that can be adjusted according to network changes. Algorithm 4.1 is the pseudo code of the distributed learning mechanism.

Algorithm 4.1 CMT-SCTP learning algorithm

1: action set $=\{decWin(), incWin(), stay()\}$
2: inputs $= \{RTT_i, cwnd_i, RTO_i, FL_i\}$
3: output $= \{a_i \in action\ set\}$
4: **while** have packets to send at each decision epoch **do**
5: get RTT_i, $cwnd_i$, RTO_i, FL_i for all interfaces
6: $cwnd = \Sigma_i cwnd_i$
7: $RTT = \frac{\Sigma_i^n RTT_i}{n}$
8: $FL = \Sigma_i FL_i$
9: $RTO = \frac{\Sigma_i^n RTO_i}{n}$
10: determine the state of the MDP, s_t, using the Bayes classifier
11: calculate the reward r_t
12: update the Q matrix using Eq. (5.2)
13: choose the action with maximum Q value
14: inform the transport layer to change the maximum congestion window size
15: **end while**

As time passes, the learning agent develops a memory of all the events and creates a map of actions; therefore, any changes in the environment can be handled instantaneously by the agent.

4.3.4 Architecture

Each CMT-SCTP source is equipped with a Q-learning agent. The Q-learning agent sits in the transport layer. At each decision epoch, the transport layer provides the Q-learning agent with its statistics. The agent uses the Bayes classifier to determine if the source node is fair within the decision epoch. Based on the state of the system, Q-learning agent chooses an action and informs the transport layer about the action. In the next decision epoch, the Q-learning agent receives the new statistics which reflects the effectiveness of the taken action. The Q-learning agent uses the outcome to bestow a reward or penalty

to the action and stores the state-action reward in its memory for future use. The cycle continues until the Q-learning agent calculates all state-action transition probabilities.

To provide the Bayes classifier with training data, the agent inside each source node starts to collect data as soon as the node starts transmitting. Because all flows within the wireless domain pass through the gateway to access the rest of the network, the gateway has almost an inclusive knowledge of the flow rates within the wireless domain; therefore, it can determine if a node is being fair to other nodes or not based on its share of network bandwidth. At the beginning of the learning process, the gateway monitors the average throughput of incoming flows and labels the flows as fair or unfair using Jain's fairness index in Eq. (4.1) and sets a bit in the shut down message of the transport layer. The shut down message in a CMT-SCTP association is used to notify the sender that the destination is closing the association and does not accept any new packets. The source node, on the other hand, monitors the average congestion window size, RTT, RTO, and the flight size on all interfaces and labels the collected data using the set bit in the shut down message. After collecting n data points, the Bayes classifier uses the data set for training and there is no need for the gateway shutdown message feedback. The Q-learning agent starts its learning after the Bayes classifier has enough data point to classify new observations. The above design does not add any overhead to the network while providing each node with an accurate correlation between its local parameters and the global fairness index. We use Bayes classifier because it is highly scalable and the empirical results shows that it performs surprisingly well in many domains even the ones containing clear attribute dependences [113].

4.4 Evaluation

To evaluate the performance of Q-learning CMT-SCTP, we implemented the Q-learning mechanism as a Python suite, based on the algorithm 4.1. The Python suite communicates with transport layer congestion mechanism via action functions (Section 4.2) to tune the maximum congestion window damp based on the network dynamics in real-time. The current and new maximum congestion window sizes are accessible and tunable via socket functions. In our approach, each CMT-SCTP source is equipped with a learning agent (Python suite). The learning agent monitors the transport layer dynamics in real-time and determines the state of the MDP in each decision epoch. Based on the current state of the system, the learning agent decides to take an action via changing CMT-SCTP maximum window size. To find the optimum maximum window size for CMT-SCTP, the learning agent uses a reward function which is implemented as a Python suit. As such the whole agent sits on top of the Kernel and communicates with the Kernel regarding the necessary actions.s We chose to implement the Q-learning agent on top of the Kernel to facilitated integration of Q-learning SCTP into standard SCTP on any machine. The current design does not require any changes to the Kernel which normally exists within the manufacturer setting; however, it can be added on top of the manufacturer setting for any machine.

We used the same scenarios as in Section 4.2 to evaluate the performance of our learning mechanism. We collected 1000 data points as our training set to train the Bayes classifier and a test set of 1000 points to check the sanity of our classifier. The prediction accuracy of the Bayes classifier is 79%.

To evaluate the performance of our distributed fairness mechanism, we collected data over the scenarios presented in Table 4.2. Moreover, we compared the performance against the standard CMT-SCTP, and CMT/RP-SCTP in scenario 9. To investigate the effec-

scenario	description
1	TCP on node A, standard SCTP on node B
2	TCP on node A, standard CMT-SCTP on node B
3	SCTP on node A, standard SCTP on node B
4	SCTP on node A, standard CMT-SCTP on node B
5	TCP on node A, Q-learning SCTP on node B
6	TCP on node A, Q-learning CMT-SCTP on node B
7	SCTP on node A, Q-learning SCTP on node B
8	SCTP on node A, Q-learning CMT-SCTP on node B
9	SCTP on node A, CMT/RP-SCTP on node B

Table 4.2: Evaluation scenarios

tiveness of our proposal on the standard SCTP, we implemented the Q-learning fairness mechanism in single-homed SCTP as well and measured the performance of Q-learning SCTP against standard SCTP and TCP in scenarios 5 and 7. The measurement results from scenarios 5 and 7 demonstrate that our proposed fairness mechanism can be adopted to other transport layer protocols.

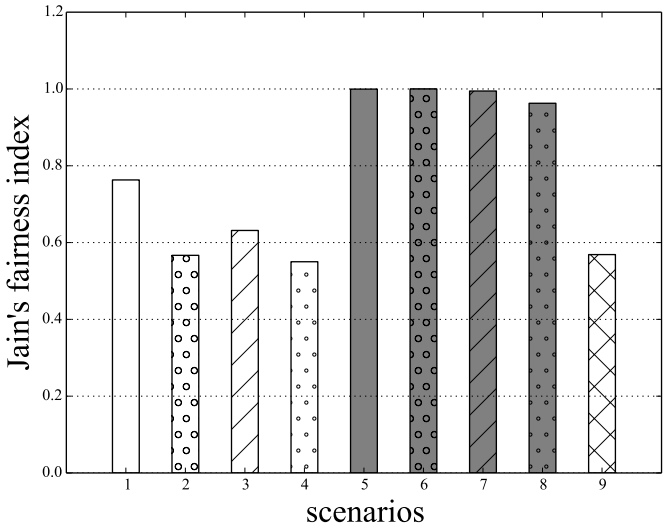


Figure 4.5: The Jain's fairness index of Q-learning SCTP and CMT-SCTP in comparison with standard SCTP and CMT-SCTP in different scenarios of Table 4.2

Figure 4.5 shows that the Q-learning mechanism boosts the fairness index drastically, i.e., the Jain's fairness index of the network rises very close to 1 in scenarios in which Q-learning CMT-SCTP or Q-learning SCTP is used on node B (the gray bars, scenarios 5, 6, 7, and 8). The dynamic damp on congestion window size does not interfere with congestion

control mechanism of SCTP; instead, it creates a virtual limit for the available bandwidth of the source node. Therefore, the dynamic damp controls the number of packets poured into the channel by slowing down the transport layer and provides other nodes with more opportunities for transmission. Following a decrease in the number of packets poured to the channel by an aggressive node, nodes located in farther away hops from the gateway start to utilize the channel more efficiently.

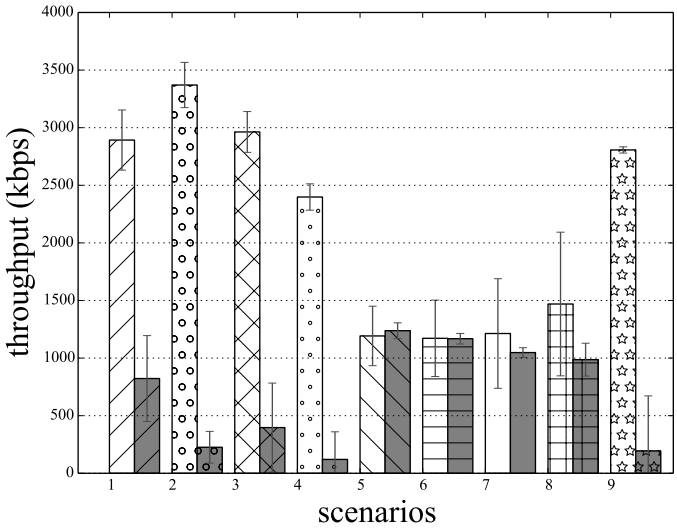


Figure 4.6: Average throughput of different scenarios (kbps) in different scenarios of Table 4.2. Scenarios 1,2,3 and 4 are the standard SCTP and CMT-SCTP, while scenarios 5, 6, 7, and 8 are the Q-learning SCTP and CMT-SCTP. Scenario 9 shows the measurement results of CMT/RP-SCTP.

Figure 4.6 depicts the throughput of the flow coming from node A (bars with the white background) as compared to the flow coming from node B (bars with the gray background) in different scenarios. The powerful effect of Q-learning CMT-SCTP in suppressing the aggressive behavior of the standard CMT-SCTP is obvious when comparing scenario 2 with 6 in Figure 4.6. Although the average throughout of CMT-SCTP on node B decreases in scenario 6, the throughput of node A increases drastically (5 times as compared to scenario 2). The same trend can be seen for scenario 4 vs. 8 where node B compromises throughput

to contribute to network fairness. Both scenarios 6 and 8 reveal the effectiveness of our algorithm in a real life wireless multi-hop setting and in presence of interferences and other sources of noise.

We used our algorithm with single-homed SCTP as well to investigate the effect of our fairness mechanism on SCTP. The measurements in Figure 4.6 show that the distributed fairness mechanism is in fact effective even in single-homed SCTP (scenario 1 vs. scenario 5 and scenario 3 vs. scenario 7).

4.5 Discussion and Comparison

To propose an effective fairness solution for CMT-SCTP over wireless multi-hop networks, one has to consider the dynamic nature of the environment as an important design factor. That is, a dynamic solution that changes strategy based on the network condition is required. Therefore, the effective solution requires two characteristics: (a) monitoring/learning network conditions, and (b) choosing the correct strategy based on the perceived condition. Reinforcement learning methods meet the design characteristics in (a) and (b). Among various reinforcement learning methods, Q-learning fits our needs the best, as it is a model-free technique. Q-learning can be used to find an optimal strategy-selection policy for any given finite state Markov decision process. The above reasoning justifies our choice of the fairness solution for CMT-SCTP over wireless multi-hop settings.

The main purpose of Q-learning in our proposed mechanism is to decide which action to take in the next time slot, i.e., increase, decrease or maintain the maximum congestion window size. Note that the best application of Q-learning is to learn action-reward functions for stationary settings, which can be proved to converge. It is true that Q-learning can still get results in the non-stationary environment, such as wireless settings, but the Q-learning agent will take more time to be aware of the changes. Due to the time-varying

network conditions, sometimes rapidly, the stationary assumption cannot always hold and it can make Q-learning less suitable for wireless networks. However, there are ways to make sure that the convergence rate stays within an acceptable range for dynamic environments. Using a learning rate of $\alpha = \frac{1}{(1+t)^{0.77}}$ brings down the convergence rate to a polynomial in $\frac{1}{(1-\gamma)}$, where γ is the discount factor [90]. We use a learning rate of $\alpha = \frac{1}{(1+t)^{0.77}}$ to ensure that our proposal complies with the dynamic nature of the environment.

Another consideration while using Q-learning for any scenario is the computational overhead. Assuming that in a specific scenario action a_i alleviates the aggressive behavior of a specific flow while keeping the throughput at its maximum possible, and τ iteration is needed before the algorithm converges, then, the overhead of the action to the learning node is:

$$Overhead = \frac{1}{2}\Sigma_{t=1}^{\tau}\Sigma_{i=1,i\neq j}^{3}P_i(t)O(a_i) \tag{4.3}$$

where $P_i(t)$ is the probability of choosing action i at iteration t and depends on the values in the Q matrix, and the reward function. $O(a_i)$ is the overhead of performing action a_i and τ is the convergence time. Based on [90], in a Q-learning scenario with a polynomial learning rate, the convergence time depends on covering time L. Covering time indicates the number of iterations needed to visit all action-state pairs with the probability of at least 0.5 starting from any pair. The convergence time τ depends on covering time with the order of $\Omega(L^{2+\frac{1}{\omega}} + L^{\frac{1}{1-\omega}})$ with the smallest amount at $\omega = 0.77$. In our experiments, the covering time is tractable as we have only 4 states and in each state we have 3 actions. Therefore, the action-state pair space is as large as 81 which leads to a small covering time and a small convergence time τ. Small convergence time, τ, decreases the accumulation of terms in equation (4.3) and thus leads to a small overhead.

One of the great advantages of our proposal is the ability to support different levels of service. Our proposal can offer flexibility to service providers for setting a minimum or a

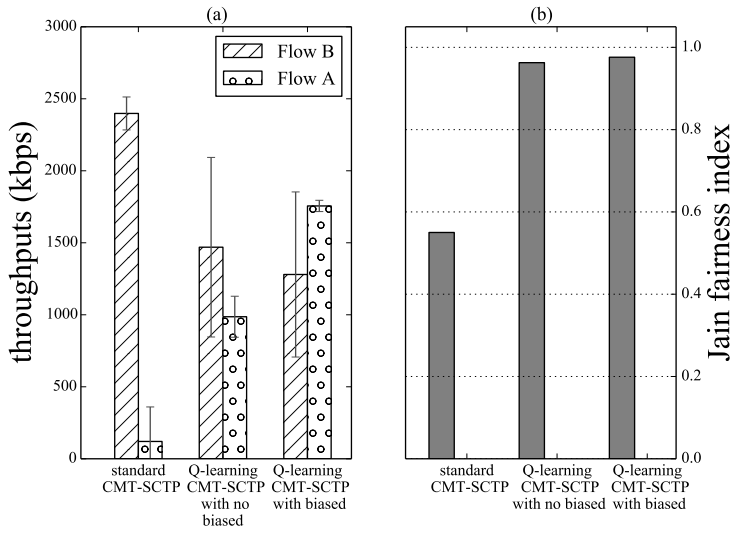


Figure 4.7: Average throughput of flows with different level of QoS using Q-learning fairness mechanism.

maximum for the dynamic damp, and to create different levels of QoS for costumers based on their demand. To show the flexibility of our Q-learning fairness mechanism in offering different levels of QoS to costumers who are willing to pay more to gain a bigger share of bandwidth, we set the maximum congestion window size of node B in favor of node A and evaluate the performance of our proposed mechanism by measuring the average throughput. The result confirms our claim that the Q-learning mechanism is capable of providing different levels of QoS and can be biased towards different flows. In Figure 4.7, the middle bar shows the result of applying the Q-learning mechanism without any bias towards any node, the first bar from the left shows the measurement results of using no fairness mechanism and the third bar from the left reports the effect of a biased Q-learning mechanism. Tuning the coefficient in the reward function and also assigning limits to the dynamic damp in the congestion control mechanism can give the provider the flexibility of offering different amounts of bandwidth share. Figure 4.7 shows that even though the Q-learning agent is suppressing source B in favor of node A, Jain's fairness index remains in the desirable range.

flavors of CMT-SCTP	link-centric fairness	network-centric fairness
CMT-QA [75]	not fair in loaded wireless scenarios	not explored
Q-learning CMT	yes	yes
CMT/RP-SCTP [18]	yes	is not explored

Table 4.3: Brief comparison on different flavors of CMT-SCTP

A brief comparison of available CMT-SCTP flavors has been provided in Table 4.3. Network-centric fairness is measured using the Jain's fairness index in Eq. (5.1) and link-centric fairness is a local measure of fairness over a bottleneck link. Table 4.3 summarizes the comparison of Q-learning CMT-SCTP with other available mechanisms. Although CMT/RP-SCTP increases the fairness of CMT-SCTP over a shared bottleneck, the results of the study of [18] were collected in a highly controlled simulated environment and over a very simple topology, offering no insight on the behavior of CMT-SCTP in more realistic settings or wireless networks. Besides, the algorithm offers the service provider with no say on bandwidth allocation, which is crucial when dealing with scenarios in which the costumer pays more to access to a higher share of bandwidth. As stated in Table 4.3, Q-learning CMT-SCTP offers more in terms of both fairness and throughput. Q-learning CMT-SCTP only needs to be implemented at the sender side and does not require any changes in the infrastructure. Further, it gives the provider the ability to offer different levels of QoS to costumers based on their demand, making it a lucrative candidate for the transport layer of multihomed devices.

4.6 Conclusion

We investigated the fairness behavior of CMT-SCTP against non-CMT flows (both TCP and SCTP) coming from nodes farther away from the gateway over a testbed in an office

area network. Our measurements showed that CMT-SCTP is highly aggressive towards single-homed flows coming from farther away hops. We proposed a distributed fairness mechanism that uses Q-learning to tune the maximum congestion window size of the CMT-SCTP association based on the statistics of transport layer to alleviate the aggressive behavior. We implemented our mechanism in FreeBSD over raspberry pi and tested the effectiveness of our mechanism in a testbed of raspberry pi's. Our measurements proved that the proposed Q-learning fairness mechanism improves the Jain fairness index up to 30% which is a significant increase of average throughput for a starving flow. Moreover, we used our Q-learning mechanism on single-homed SCTP to investigate whether the fairness mechanism has any effect on fairness of single-home SCTP against other flows coming from farther away hops. Our measurements showed that the Q-learning mechanism effectively increases the fairness of single-homed SCTP towards farther away nodes. The proposed mechanism needs to be only implemented at the sender side and does not require any changes in the infrastructure. Moreover, it gives the provider the ability to offer different levels of QoS to costumers based on their demand which makes it a lucrative candidate.

Chapter 5

How Network Monitoring and Reinforcement Learning Can Improve TCP Fairness in Wireless Multi-Hop Networks

In this chapter, we aim to adopt the fairness mechanism designed in Chapter 4 to TCP. Our proposal uses a distributed mechanism to monitor the network anomalies in resource allocation and tune TCP parameters accordingly. Each TCP source models the state of the system as an MDP and uses Q-learning to learn the transition probabilities of the proposed MDP based on the observed variables. To maximize TCP fairness, each node hosting a TCP source takes actions according to the recommendations of the Q-learning algorithm and adjusts TCP parameters autonomously. Our algorithm preserves autonomy of each node in decision making process and does not require a central control mechanism or control message exchange among nodes. Unlike the existing machine learning solutions, i.e. TCP ex Machina, our proposal is compatible with the computational capacity of the current infrastructure. We call our approach Q-learning TCP. The contributions of this chapter can be summarized as:

- Modeling the multi-hop network as an MDP in each TCP source and using Q-learning algorithm to monitor and learn the dynamics of the network and the proposed MDP.
- Finding a cross-layer distributed and scalable solution for TCP fairness over multi-hop networks with no extra overhead. Our proposal enhances TCP fairness over multi-hop networks in favor of flows traversing a longer number of hops with negli-

gible impact on flows with a shorter number of hops via changing TCP parameters cooperatively based on the recommendation of the Q-learning algorithm

- Enhancing TCP fairness by a factor of 10% to 20% without any feedback messaging and no incurred overhead to the medium

The rest of this chapter is organized as follows: Section 5.1 is a detailed description of our algorithm followed by the implementation specifics in Section 5.2. Performance evaluation of our proposed algorithm is presented in Section 5.3 via extensive simulation and testbed experimentation. A discussion on implications of Q-learning TCP along with a comparison with available fairness techniques is presented in Section 5.4. Section 5.5 concludes this chapter.

5.1 Q-learning TCP Architecture

In our approach, each TCP source is equipped with a Q-learning agent that sees the world as an MDP. In each decision epoch, the agent receives network statistics in the form of state space variables. The agent uses the received information to determine the state of the MDP; then, the agent takes an action via fine tuning TCP parameters. In the next decision epoch, the environment responds with the new state. The learning agent uses a reward function to receive feedback on the consequences of the taken action on TCP fairness. The learning process continues as long as the network is up and running; it basically works as a memory that can be adjusted according to network changes.

In the following sub-sections, we present a detailed overview of the key factors of Q-learning TCP including states, action set, reward function, and transition probabilities.

5.1.1 States

The state space of our proposed Q-learning algorithm in each TCP source is in the form of $S = (fairness\ index, aggressiveness\ index)$. To measure fairness index in each decision interval, the agent uses Jain's fairness index as in equation (5.1) [111]:

$$J_k^t(x_1, x_2, ..., x_n) = \frac{(\Sigma_{i=1}^{i=n} x_i)^2}{n \times \Sigma_{i=1}^{i=n} x_i^2}, \tag{5.1}$$

where x_i is the data rate of flow i, n is the number of flows that are originated from or forwarded by node k, and J_k^t is the Jain's fairness index at node k at decision epoch t. The Jain's fairness index is a continuous number that varies between 0 and 1; with 0 the worst fairness index and 1 an absolute best fairness condition. To tailor the Jain's fairness for a discrete state space, we divided the $[0, 1]$ interval to p sub-intervals $[0, f_1], (f_1, f_2], ..., (f_{p-1}, 1]$. Instead of using a continuous fairness index, we quantize it to have manageable number of states. Number of states is important in convergence of the learning algorithm.

The aggressiveness of each TCP source in each decision epoch is measured as in equation (5.2):

$$G(i) = \frac{number\ of\ packets\ originated\ from\ node\ i}{total\ number\ of\ packets\ forwarded\ by\ node\ i}. \tag{5.2}$$

The aggressiveness index is a continuous amount that varies between 0 and 1. To tailor the aggressiveness index for discrete state space, we divided the $[0, 1]$ interval to q sub-intervals $[0, g_1], (g_1, g_2], ..., (g_{q-1}, 1]$. As such, the state space of the MDP is in the form of equation (5.3) with the size of $p \times q$.

$$S = \{(f_t, g_t) | f_t \in \{0, f_1, ..., f_p\}\ and\ g_t \in \{0, g_1, ..., g_q\}\} \tag{5.3}$$

Choosing a suitable value for p and q is a critical task. A small p or q shrinks the state

space and positively affects the convergence rate; however, larger quantization intervals disturb the transparency of the changes in the system to the reward function. Reward function uses the state of the system as a decision criterion to reward or penalize the latest $(state, action)$ pair. Our extensive simulations and testbed experimentation show that choosing $3 \leq q \leq 4$ and $3 \leq p \leq 4$ provides the Q-learning TCP with enough number of states to significantly increase the fairness index of the network and convergence rate.

Fairness index obviously is a good indicator of how well the Q-learning TCP is doing in terms of enhancing the fairness. However, fairness index is not enough to make a decision on fairness of the TCP source. Therefore, we define aggressiveness index to indicate if a TCP source is fair to other TCP sources or not. The aggressiveness index calculates the share of the TCP source located on a specific node in the outgoing throughput of the node. A high aggressiveness index along with a low fairness index in a TCP source triggers the learning agent to make changes to the TCP parameters and to force the TCP source to be more hospitable towards other flows. The desirable state for the learning agent is the state with a high fairness index and an aggressiveness index of T_{fair}. T_{fair} is the fairness threshold of the node. The fairness threshold depends on the number of flows originating and passing through the node and the priority of each flow. As an example, if 3 flows are passing via a node, and 2 other flows are originating from the node, assuming the same priority for all 5 flows, the fair share of the node from network resources and the fairness threshold is $\frac{2}{5}$. Any aggressiveness index above fairness threshold is an indication of the unfair resource allocation by the TCP source on the node.

Both fairness index and aggressiveness index are calculated based on the number of packets received or transmitted in each decision epoch in the TCP source. As such, both variables are accessible in each node and there is no need to get a feedback from other nodes. Let's emphasize that the objective of our MDP is to enhance TCP fairness co-

operatively and accomplish this objective via moving towards the goal state; therefore, choosing fairness index and aggressiveness index as state variables of our MDP is justified.

5.1.2 Action set

The Q-learning agent uses the action set to constrain TCP aggressive behavior. Findings of [114] suggested that the maximum TCP congestion window size plays a crucial role in the aggressive behavior of TCP. However, putting a static clamp on TCP congestion window size might cause an under-utilization of the network resources. Therefore, to avoid any under/over-utilization of network resources, we use a dynamic clamp on TCP congestion window size. The learning agent dynamically changes the maximum congestion window size of TCP without interfering with the congestion control mechanism via the action set. As such, TCP uses the standard congestion control mechanism along with a dynamic clamp on the congestion window to limit any aggressive window increase. The agent uses the action functions in Algorithm 5.1 to change TCP parameters in each decision epoch.

Algorithm 5.1 Q-learning action set

```
1: if action = decrease then
2:     δ = 50%
3:     decreases the TCP maximum window size by δ
4: else if action = increase then
5:     increases the TCP maximum window size by δ
6: else
7:     no change to maximum congestion window size
8: end if
```

The Q-learning TCP does not interfere with the congestion control mechanism of TCP, only changes the maximum TCP window size; the maximum window size can be decreased up to slow start threshold. In our algorithm, we used δ as 50% of the latest increase/decrease of the current TCP maximum window size. The above action set, which resembles a binary search behavior, serves perfectly in a dynamic environment. At the be-

ginning of the learning process, the maximum congestion window size is set either to 65536 bytes [115] or the amount allowed by the system. The learning agent starts searching for the optimized maximum TCP window size by halving the current maximum TCP window size. As such, the search space decreases into half. The agent chooses either half randomly due to the random nature of the search algorithm in the beginning of the learning process. The agent starts swiping the search space using the *decrease*( ), *increase*( ), and *stay*( ) functions and uses the reward function as the guide to the optimum state. During the learning process, the agent develops a memory and uses its memory after convergence as a series of policies to handle any changes in the dynamics of the system. As a result, in any state, the learning agent knows how to find its way to the optimum clamp. The Q-learning agent converges when all the available action series and their associated reward are discovered.

5.1.3 Transition probabilities

In our proposal, the state space is in the form of $S = (fairness\ index, aggressiveness\ index)$. Both fairness and aggressiveness indices depend on the number of packets transmitted or received in the most recent decision epoch. Therefore, any state transition only depends on the latest state of the system. As such, all states are Markovian. We use Q-learning to obtain the transition probabilities of the proposed MDP.

5.1.4 Reward function

According to [93], an efficient reward function should have the following conditions:

- the reward function has a uniform distribution for states far from the goal state.
- the reward function points in the direction of the greatest rate of increase of reward in a zone around the goal state.

The reward function that we use for our model is in the form of equation (5.4), which is a summation of fairness reward function and network utilization reward function.

$$R(s,a) = \beta_u e^{-\frac{d(u_s,u_{s*})^2}{2\sigma_u^2}} + \beta_f e^{-\frac{d(fs,f_{s*})^2}{2\sigma_f^2}} \tag{5.4}$$

u_s and u_{s^*} are the the network utilization factor in states s and s^*. Network utilization factor is the accumulative throughput of all the incoming and outgoing flows in a node. f_s and f_{s^*} are the Jain's fairness index in states s and s^*. s^* is the goal state.

There is always a trade off between fairness and the throughput/efficiency of the network. In a highly effective network, the utility function is focused on maximizing the aggregated throughput of the network which might not be optimized based on fairness. In our scheme, we optimize TCP performance based on both fairness and throughput. To create a balance between the two, we use the aggressiveness index (throughput control factor) and fairness index (fairness control factor). The aggressive nodes have to compromise the throughput in favor of starved nodes to increase the fairness index of the network. In our mechanism, the reward function includes both fairness and throughput. One has to keep in mind that optimizing TCP performance based on both fairness and throughput results in some compromise on throughput to increase the fairness. Moreover, Quality of service (QoS) can be added to our scheme via the reward function. Putting different weights on either throughput or fairness via changing the coefficients in reward function (β_u,β_f,σ_u,σ_f) can result in different levels of QoS.

5.2 Integration of Q-learning Agent and TCP

As depicted in algorithm 5.2, the MAC layer collects the number of sent packets of each flow passing through the node and sends them to the learning agent which is located in

TCP source every t seconds (t is the length of decision epoch). The learning agent uses the latest information to calculate the aggressiveness and fairness index and determines the current state of the system. The reward function module uses the current state of the system, previous state of the system, and the latest action to calculate the immediate reward of the agent based on the latest $(state, action)$ pair. When the agent obtains the immediate reward, it updates the Q-matrix based on the Ballard equation (See 2.2.3). Finally, the agent chooses an action based on the recent Q-matrix and informs TCP to adjust its maximum window size based on the chosen action. We are using a delayed reward system because the agent has to wait for the system to settle down in a specific state to figure out the instant reward.

Algorithm 5.2 Q-learning TCP algorithm of node i

```
1: action set ={decrease,increase,stay}
2: absorbing state = (f_goal, g_goal)
3: while not in goal state do
4:    for every t seconds do
5:       get the number of sent packets by node i
6:       for each flow being forwarded by node i do
7:          get the number of sent packets
8:       end for
9:       calculate the fairness index f_t, based on (5.1)
10:      calculate the aggressiveness index g_t, based on (5.2)
11:      determine the state according to (5.3)
12:      calculate the reward r_t based on (5.4)
13:      update the Q matrix according to (2.1)
14:      choose the action with maximum Q value
15:      take the action (inform TCP)
16:   end for
17: end while
```

5.3 Performance Evaluations

Q-learning TCP is specifically aimed for the wireless mesh network environment; as such, all the simulations and testbed are designed to present the interaction of TCP and the unique

characteristics of the wireless mesh setting. In this section, we present the numerical results of our proposed method and demonstrate the effectiveness of our fairness mechanism as compared to TCP, TCP-AP, and TCP ex Machina.

We first evaluate the performance of the Q-learning TCP in a multi-hop setting in which all the nodes are located in the same wireless domain and participate in the optimization process in section 5.3.1. Section 5.3.3 presents the performance of Q-learning TCP over a testbed with real data in an office environment.

5.3.1 Chain topology

The chain topology is a good scenario to evaluate the effectiveness of Q-learning TCP over Wireless mesh networks (WMN), because it can create a competitive environment for flows with different number of hops which is the main feature of each wireless mesh topology. We use Jain's fairness index and the flow throughput as the comparison metric parameters. We set up a multi-hop network of 3 nodes located in a chain topology with 150 meters spacing between neighboring nodes, as shown in Figure 5.1. Each node is equipped with a

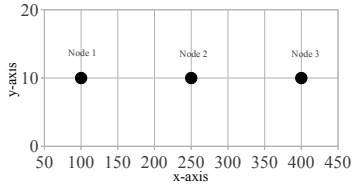


Figure 5.1: Chain topology - 3 nodes, 2 flows

802.11*b* network interface. Nodes 1 and 2 are equipped with an FTP source that transmits packets to node 3. The transmission range is around 250 meters, and the carrier sensing range is around 550 meters for each wireless interface. The data rate for IEEE 802.11*b* is 2 Mbps and the RTS/CTS mechanism is on. Each TCP packet carries 1024 bytes of application data.

For the Q-learning scheme, we had to determine the state space of the system and the

reward function. As mentioned in section 5.1, the fairness index and aggressiveness index have values between 0 and 1. For fairness index, we divided the $[0, 1]$ interval into four sub-intervals, $f = \{[0, 0.5), [0.5, 0.8), [0.8, 0.95), [0.95, 1]\}$. For the aggressiveness index, we split $[0, 1]$ interval into three sub-intervals, $g = \{[0, 0.5), [0.5, 0.8), [0.8, 1]\}$. We can split both the fairness index and aggressiveness index into smaller sub-intervals and provide more control for the learning agent to tune TCP parameters for more desirable results. Although having smaller intervals facilitates the learning agent decision making, an increase in number of intervals increases the state space size and slows down the learning process convergence rate. After the quantization, the state space reduces to (5.5):

$$s = \{(f, g)|f = \{0, 0.5, 0.8, 0.95\}\ , g = \{0, 0.5, 0.8\}\} \tag{5.5}$$

The above state space provides each node in the network with a realistic understanding of the resource allocation of the neighboring nodes. The immediate reward function that we used for our Q-learning TCP is in the form of equation (5.4). We ran each simulation for 30 times for 95% confidence interval. The length of each simulation is 1000 seconds.

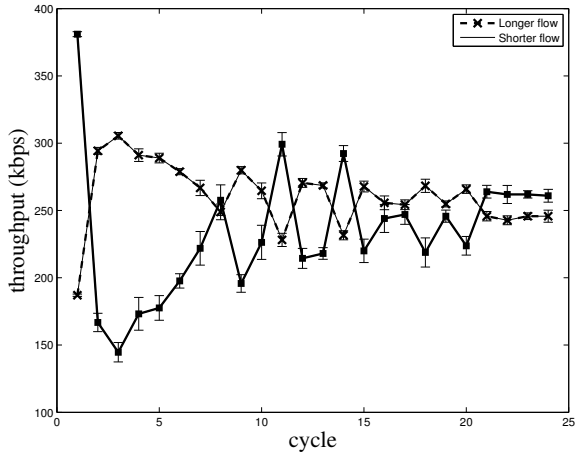


Figure 5.2: Throughput changes of the flows in the chain topology of Figure 5.1. Each cycle is equivalent of 40 seconds

Figure 5.2 shows the throughput changes during the learning process. At the beginning of the learning process, the Q values are all zero; therefore, the agent starts a systematic search to determine the effect of each action on the state of the system. Because we choose a Gaussian reward function, the Q-learning agent gradually moves to states adjacent to the goal state. The systematic search behavior exists during the simulation, but the range of the search circle diminishes as the learning process converges to the goal state. As depicted in Figure 5.2, at the beginning of the learning process, the throughput of both flows fluctuates. As time goes by, the fluctuation of both flows dwindles to negligible amount. As the learning process progresses, the agent visits each state sufficient number of times to find the best policy (the best maximum TCP window size) to maximize its acquired rewards. Eventually the learning agent in node 2 converges to $s = (0.95, 0.5)$, where 0.95 is the fairness index and 0.5 is the aggressiveness index.

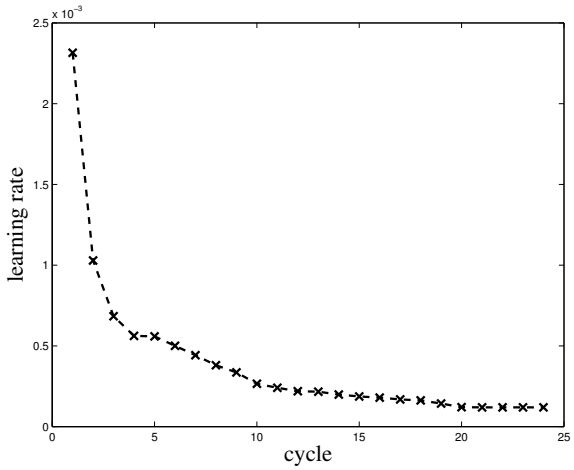


Figure 5.3: Learning rate (proof of convergence) in topology of Figure 5.1. Each learning cycle consists of 40 seconds.

To investigate the convergence of the learning algorithm, we calculated the average learning rate, as shown in Figure 5.3. The average learning rate of the process is calculated as $\frac{1}{(E\{n(s,a)\})}$, where $E\{n(s,a)\}$ is the average number of times each $(state, action)$ pair visited by the agent. According to [89], a deceasing average learning rate is an indication

of the Q-learning convergence process. Figure 5.3 shows that the learning rate approaches 0 which guarantees the convergence of the learning algorithm.

Figure 5.4 shows the changes of network utilization factor as the learning progress. As depicted in Figure 5.4, the network utility factor fluctuates widely at the beginning of the learning process. However, the network utility factor settles to a value within the desirable range when the learning process converges.

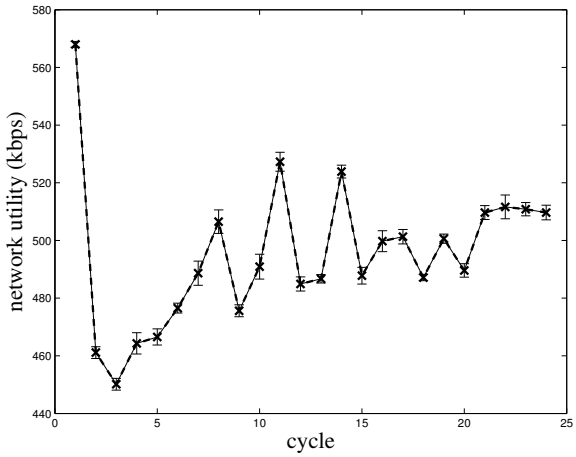


Figure 5.4: Network utilization factor (kbytes/sec) for the topology shown in Figure 5.1. Each learning cycle consists of 40 seconds.

As the learning process converges to the desirable state, the Jain's fairness index of the network settles and the fluctuations become negligible, as demonstrated in Figure 5.5.

We compared our scheme with TCP-AP [51] and TCP ex Machina [68]. We implemented TCP-AP based on the algorithm in [51]. In TCP-AP, the sending rate is limited based on the changes in RTT.

Figure 5.6 graphs the performance of TCP-AP, TCP, Q-learning TCP, and TCP ex Machina. TCP-AP performs closely to our learning method in terms of the fairness index of the network. However, the fair resource allocation in TCP-AP comes at the cost of network utility. The drop in the network utility factor in TCP-AP is caused by the frequent pauses in data transmission. TCP ex Machina, on the other hand, performs similar to standard

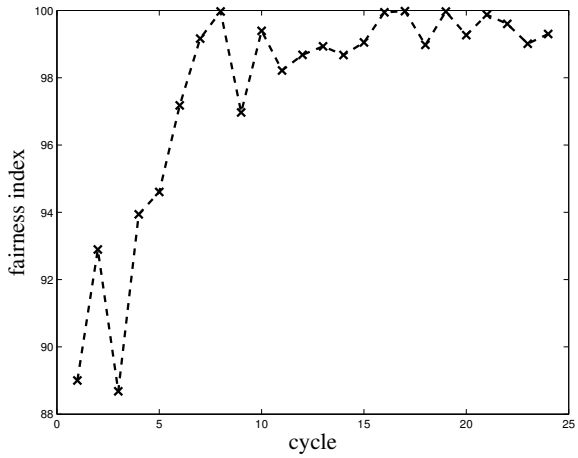


Figure 5.5: Jain's fairness index in the chain topology of Figure 5.1. Each cycle is equivalent of 40 seconds

TCP; the reason behind this behavior of TCP ex Machina over the multi-hop wireless setting is that the learning mechanism in TCP ex Machina is optimized for wired settings and the re-learning process requires a great deal of computational resources which almost is impossible to be done on the current wireless nodes within a reasonable amount of time. Table 5.3.1 summarizes the comparison of Q-learning with TCP-AP and TCP ex Machina.

Table 5.1: Network metrics parameters for different TCP variations of Figure 5.1

TCP variation	Jain's fairness index	Network utility
Legacy TCP	89%	509 Kbps
TCP-AP	99%	83 Kbps
TCP ex Machina	84%	694 Kbps
Q-learning TCP	99%	468 Kbps

5.3.2 Larger scale WMN

To evaluate our proposed algorithm further with non-static traffic pattern and a random mesh topology, we generated a random topology in ns2, as illustrated in Figure 5.7. There exist 6 flows in the network; all flows are destined towards node 0 and are originated from

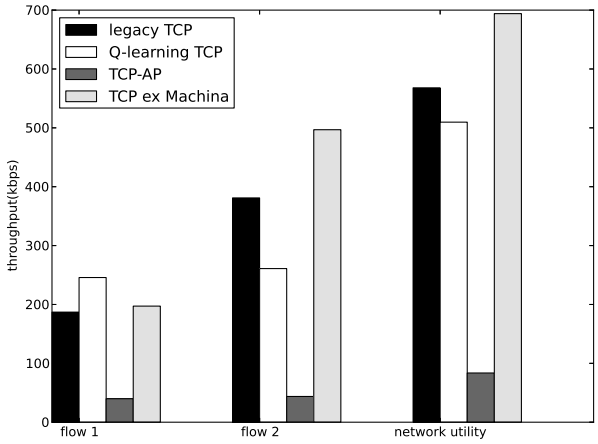


Figure 5.6: Performance comparison between legacy TCP, Q-learning TCP, TCP-AP, and TCP ex Machina

nodes 8, 3, 5, 7, 1, and 4. To study the performance of Q-learning TCP under non-static data traffic, we programmed all the sources to generate data for random length intervals and intermittently.

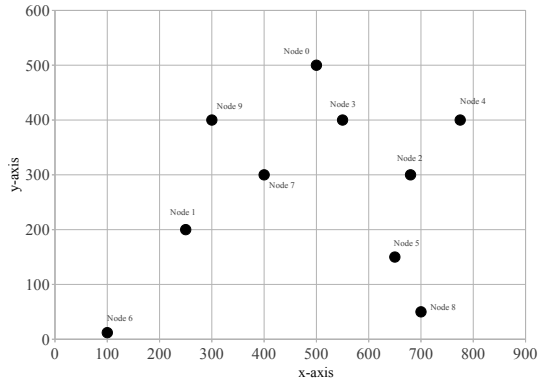


Figure 5.7: Random network topology - 10 nodes

As depicted in Figure 5.8, the resource allocation in the legacy TCP is severely unfair as nodes 8, 1, and 4 are starved while nodes 3 and 7 aggressively consume the bandwidth. However, the Q-learning TCP pushes node 3 to decrease its network share and provide more transmission chance for other nodes. The learning agent of node 5 also indicates

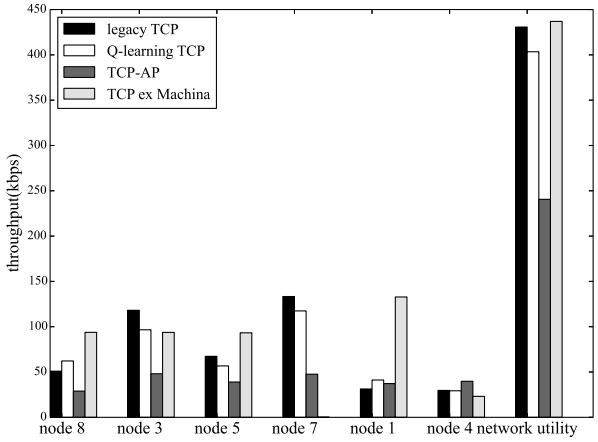


Figure 5.8: TCP flow throughput and network utility in kbytes/sec for scenario of Figure 5.7

an unfair resource allocation and forces node 5 to slow down the data sending rate. On the other hand, node 8 experiences an undergrowth of the congestion window size and starts to increase the sending rate as node 5 decreases the sending rate. Node 3 and node 5 cooperatively provide other nodes with more sending opportunities by decreasing the sending rate which results in an increase in node 8's sending rate. On the other side of the network, node 7 consumes a bigger portion of the bandwidth as compared to node 1; therefore, the Q-learning TCP forces node 7 to decrease its sending rate and provide other nodes with more sending opportunities. A comparison of TCP, TCP-AP, TCP ex Machina and Q-learning TCP is presented in Table 5.3.2.

TCP-AP outperforms both TCP and Q-learning TCP in fair resource allocation in the scenario of Figure 5.7. However, the high fairness index of TCP-AP comes at the cost of drastic decrease in network utilization by a factor of 50%. The reason behind the extreme decrease in network utility factor in TCP-AP is the over-estimation of RTTs and excessive overhead caused by feedback messages. TCP ex Machina performs as poor as legacy TCP and causes one of the flows to starve completely.

To investigate the convergence of the learning process, we graphed the learning rate of

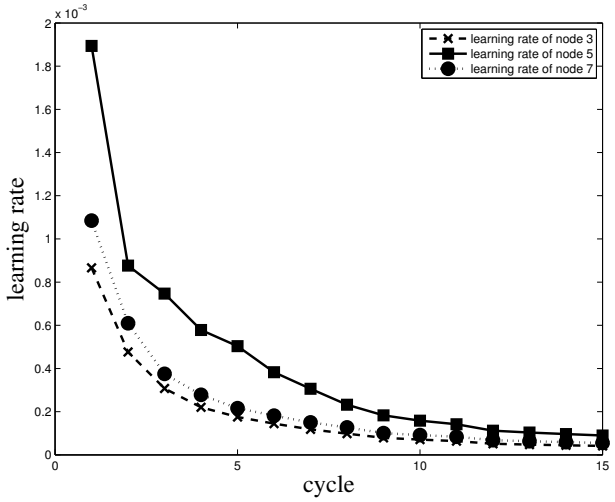


Figure 5.9: Learning rate of nodes 3, 5, and 7 in scenario of Figure 5.7

the learning process for nodes 3, 5, and 7 in Figure 5.9. The agent learning rate for all three nodes converges to 0 as the learning process progresses. The learning rate of node 3 is higher that the two other nodes because node 3 has to make more changes to get to the optimum state. More changes translate into more state transitions and consequently a higher convergence rate.

Table 5.2: Network metrics parameters for different TCP variations of scenario 5.7

TCP variation	Jain's fairness index	Network utility
Legacy TCP	75%	430 Kbps
TCP-AP	97%	240 Kbps
TCP ex Machina	71%	436 Kbps
Q-learning TCP	83%	403 Kbps

5.3.3 Testbed

To evaluate the performance of Q-learning TCP, we use the testbed setup in Section 2.3 and 2.3.2 (Figure 5.10). The data paths are as follows:

- flow A: AR_1BR_2C

- flow B: BR_2C

We implemented the Q-learning mechanism as a python suite based on algorithm 4.1 in node B and node A. The Q-learning mechanism communicates with the transport layer via action functions in Section 5.1.2 in order to tune TCP maximum congestion window size. While all the users over the network continue their day to day activity.

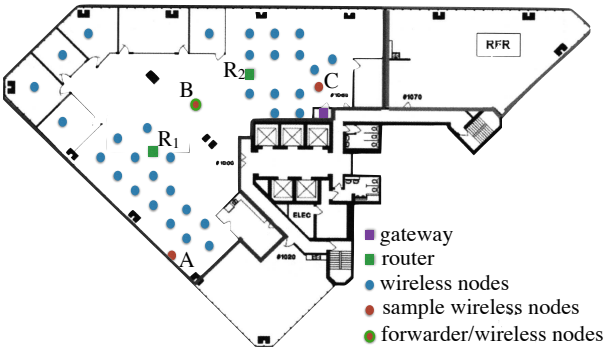


Figure 5.10: Testbed topology.

To generate real-world traffic profile over the testbed, we use findings of Brownlee *et al.* [116]. According to measurements of [116] over two real-life networks, Internet streams can be classified into two groups: dragonflies and tortoises. Dragonflies are session with lifetime of less than 2 seconds while the tortoises are the sessions with long lifetimes, normally over 15 minutes. Authors of [116] showed that although the lifetime of sessions varies; the lifetime distribution shape is the same for both Tortoise and dragonflies and it does not not experience rapid changes over time. Findings of [116] are critical to the design of Q-learning TCP and its interaction in real world; the fact that the distribution of the lifetime of the streams does not change rapidly over time fits well with the characteristics of Q-learning. Based on [116], 45% of the streams have a lifetime of less than 2 seconds, 53% have a lifetime between 2 seconds and 15 minutes, and the rest has life times more than 15 minutes (usually in the order of hours). We use these findings to generate traffic on node A and B along with the other existing day-to-day traffic within the office network.

Table 5.3 shows the result of our measurements over the testbed of Figure 5.10. TCP Q-learning outperforms TCP Reno on node A with the big margin of 85% of increase in the average throughput. Node B which acts as a source and a forwarder node has to compromise its throughput to enhance the fairness of the network. The average throughput of node B decreases by 3% to accommodate a 85% boost in throughput of node A which is a drastic increase with a minimal compromise.

Table 5.3: A comparison between Q-learning TCP and TCP Reno

TCP variation	Source	Throughput (Kbits/sec)	95% Confidence interval (Kbits/sec)
TCP Reno	node A	883	less than 100
TCP Reno	node B	4003	less than 400
Q-learning TCP	node A	1549	less than 400
Q-learning TCP	node B	3877	less than 1000

The larger confidence interval in Q-learning TCP is caused by the changes of the maximum congestion window size according to the optimal policy of Q-learning during the discovery.

5.4 Discussion and Comparison

Comparing Q-learning TCP with other well-known existing fairness methods [117][48][51], Q-learning TCP does not incur any overhead to the network with the expense of extra computation at each node. The focus of LRED [117] is on enhancing TCP throughput over WMN and fairness enhancement is not one of the design objectives. However, the pacing mechanism of LRED enhances the fairness as a side-effect at the cost of excessive transmission delay. The extra transmission delay in LRED pacing mechanism alleviates the hidden terminal issue; however, the imposed delay is fixed in size and is not adjustable to the dynamic nature of the WMN. NRED uses a dropping mechanism to decrease the

competition and provide more resources for the starved flows. However, the dropping probabilities are calculated and broadcasted constantly, thus incurring a heavy overhead to the shared medium. TCP-AP uses the received signal strength indicator (RSSI) to infer information regarding the hidden terminal issue; however, RSSI causes an over-estimation of transmission delay in its pacing mechanism and decreases the TCP throughput drastically. As such, TCP-AP still requires feedback messaging from the neighboring nodes for hidden terminal distance calculations. TCP ex Machina, another comparable mechanism, requires excessive computational resources for its congestion control mechanism optimization which is not compatible with current network infrastructure resources. Our method achieves the fairness enhancement of TCP at a cost of reasonable extra computation of the machine learning approach in each node. In WMN that the shared medium is extremely valuable, flooding the network with excessive feedback messages or under-utilizing the links with excessive non-dynamic transmission delays to enhance the fairness is not very cost efficient. However, Q-learning TCP trades the computational simplicity in each node for TCP fairness. In a mesh setting, since the mobility of each node is very minimal, increasing the computational capacity of the nodes is not very costly. A brief comparison of LRED, NRED, TCP-AP, TCP ex Machina and Q-learning TCP is presented in Table 5.4.

Note that the complexity of Q-learning TCP is polynomial with the number of states and the convergence rate is tractable with a suitable state space size as confirmed with the simulation and testbed experiments.

Our findings confirm that in a wireless multi-hop setting, each TCP source has to cooperate with others to ensure a fair share of network resources for all end-users. TCP allocates resources with the assumption of an inclusive knowledge of the network topology and a reliable permanent access to the medium for all the end-users. However, in a wireless mesh setting, the end-user knowledge of the network topology is partial and the access to

the medium is intermittent. As such, TCP needs to collect information about other nodes to compensate for the short comings of the underneath layers. The learning agent provides the TCP source with insight on existing competition for network resources from other nodes. The insight provided by the learning agent compensates for the unfair behavior of the MAC and TCP in the wireless multi-hop environment by suppressing the aggressive response of TCP. Note that Q-learning TCP inter-works well with any variation of TCP on the other end-point because the changes to the TCP protocol stack are only in the sender side and the learning mechanism does not need any feedback from the receiver. The Q-learning TCP only relies on the information collected by the learning agent.

5.5 Conclusion

We have proposed a cross-layer monitoring and learning mechanism for fair resource allocation of TCP over WMN that uses the information obtained from the MAC layer to optimize TCP parameter to enhance the end to end fairness within a network. The learning agent uses the local fairness index and the aggressiveness index of each node to decide if the node is starving or abusing the network resources. We have used a reward function to guide the learning agent in taking correct actions that eventually allows us to solve the fairness problem in a distributed manner. We have compared our learning method with legacy TCP and TCP-AP, and TCP ex Machina via extensive ns2 simulations. Simulation results have demonstrated the superiority of our proposed method within wireless networks. Moreover, we have studied the performance of Q-learning TCP in a testbed. Testbed measurements have proved that Q-learning TCP can be a great candidate for transport protocol over current wireless multi-hop networks with minimal changes only in TCP source.

Table 5.4: A comparison between Q-learning TCP and other well-known TCP solutions. The throughput and fairness enhancement of each TCP flavor is compared against the standard TCP. As such, when a decrease/increase is mentioned, it is relative to standard TCP.

fairness solution	fairness enhancement	throughput enhancement	disadvantage
LRED [117]	slight increase	5% to 30% increase	overhead caused by broadcast messages and fixed transmission delay
NRED [48]	effective increase (Jain's fairness index of 99% in a chain topology)	up to 12% increase	excessive overhead caused by broadcast messages (over 60%)
TCP-AP [51]	effective increase (Jain's fairness index of 99% in a chain topology)	drastic decrease (up to 50%)	reliance on RSSI and excessive transmission delay
TCP ex-Machina [68]	decrease (Jain's fairness index of 84% in a chain topology)	slight increase	excessive learning time and computational resource requirement
Q-learning TCP	effective increase (Jain's fairness index of 99% in a chain topology)	slight decrease	medium computational overhead

Chapter 6

Conclusions and Future Work

In this chapter, we summarize the results and highlight the contributions of this thesis. We also suggest several topics for future work.

6.1 Research Contributions

In the present thesis, we have identified two main challenges of CMT-SCTP over wireless multi-hop networks: receive buffer blocking and unfair resource allocation while competing with single-homed flows. We have proposed novel dynamic schemes using the RL algorithm to address both challenges in Chapters 3 and 4. We have applied our findings on fairness improvement to TCP to demonstrate the adaptability of our state-of-the-art proposal in Chapter 5.

- In Chapter 3, we proposed a dynamic network coding mechanism for CMT-SCTP to address receive buffer blocking in a multipath transport data transfer in a wireless multi-hop environment. The network coding helps the transport layer to eliminate the reliance of the congestion control algorithm on the packet's transmission sequence number by sending a combination of packets instead of simply forwarding them. The network coding mechanism uses redundant packets to mask the random losses of network. We used a Q-learning mechanism within the network coding algorithm to estimate the number of redundant packets based on the dynamics of the network. The Q-learning module in the design adjusts to the dynamic nature of the wireless

environment and minimizes the number of redundant packets required to mask the random losses of network. As such, the algorithm proves to be effective in alleviating the receive buffer blocking and improving the throughput of CMT-SCTP. Our coded CMT-SCTP scheme outperforms the original CMT-SCTP with a high margin (up to 62% depending on the path dissimilarities and receive buffer size).

- In Chapter 4, we proposed a distributed fairness mechanism for CMT-SCTP over wireless multi-hop environment to address the unfair behavior of CMT-SCTP against non-CMT flows coming over form farther away hops. We used Q-learning to model the environment as an MDP and find the transition probabilities for the MDP to take suitable actions and adjust the CMT-SCTP congestion window. Our proposed mechanism enhances the fairness behavior of CMT-SCTP over wireless multi-hop networks by constantly changing the maximum congestion window size on each sub-flow. The dynamic damp on the maximum congestion window size creates a co-operative resource share in CMT-SCTP and provides the starved flows with more transmission opportunities.

- In Chapter 5, we proposed a distributed fairness mechanism for TCP over wireless multi-hop environment to address the unfair behavior of TCP against flows coming from the farther away hops. The algorithm uses reinforcement learning to monitor the dynamics of the networks and fine tunes TCP parameters to enhance the fairness of the network. Our mechanism uses the dynamic damp on maximum congestion window size of TCP to alleviate the aggressive behavior of TCP against flows coming from farther away hops. Our mechanism proves to enhance the fairness index of the network drastically. The adaptation of our proposed CMT-SCTP fairness mechanism to TCP demonstrates the powerful tool of machine learning in dealing with ever-changing environment such as wireless multi-hop networks.

6.2 Suggestions For Future Work

In the following, we discus several possibilities for extension of the current work.

1. **Network coding Q-learning mechanism for heterogeneous network** In Chapter 3, the Q-learning network coding is fully explored as a potential solution for receive buffer blocking over wireless mesh networks. One of the great directions to extend the existing work is to explore the performance of the algorithm in a heterogeneous environment. Most of the time, a multi-home device is connected over two different network such as cellular and wi-fi, i.e., smart phones; as such, one has to investigate the cons and pros of using our coded mechanism over heterogeneous environments to paint a comprehensive picture of the Q-learning mechanism.

2. **Distributed fairness algorithm for backbone network:** In Chapter 4 and 5, we proposed a distributed fairness mechanism for small to medium size network before the traffic hits the backbone. Our mechanism can be deployed in the backbone with some modifications. One of the advantages of using our mechanism before traffic hits the backbone is that the algorithm has access to the source. However, if the algorithm were to be used in the backbone, the algorithm has to be equipped with cross-layer functions. Moreover, the fairness mechanism needs to be re-designed with new action functions as the backbone network does not have access to the traffic source to be able to change the the parameters of the transport layer. As such, new control parameters needs to be found that changing those parameters has a direct effect of fairness performance of the backbone nodes. Introducing feedback messages to keep the old action functions is an easy fix to the problem; however, it defies the main advantage of the distributed mechanism which is the "no feedback" policy. The proposed mechanism has great potentials for commercialization.

3. **Centralized learning vs. distributed learning:** another approach in using learning mechanisms to fine tune the performance of the transport layer is to collect the data at the node and send the collected data to a nearby central gateway/data center to do the learning off-line and then load the result on each node. However, to implement a centralized off-line approach, one has to consider the delay and the extra messaging overhead induced by the approach versus the gain in less computational overhead for each node within the network.

4. **Distributed fairness algorithm over bigger size networks:** the proposed mechanisms in Chapter 4 and 5 create computational overhead for the nodes using the algorithm. As such, when the network size and the number of flows involved in the optimization process grows, it might have a negative impact on the convergence time of the learning mechanism. One way to adjust our mechanism for larger area networks is to break down the network into smaller clusters and uses the mechanism within each cluster. Clustering the network into smaller groups decreases the convergence time while improving the fairness effectively. However, the idea needs to be fully investigated to make sure all aspects of the design is delicately considered while using the mechanism along with clustering.

Bibliography

[1] N. Arianpoo, I. Aydin, and V. C. Leung, "Network coding: A remedy for receiver buffer blocking in the concurrent multipath transfer of data over multi-hop wireless networks," in *2014 IEEE International Conference on Communications (ICC)*. IEEE, 2014, pp. 3258–3263.

[2] T. Dreibholz, M. Becke, E. P. Rathgeb, and M. Tuxen, "On the use of concurrent multipath transfer over asymmetric paths," in *2010 IEEE Global Telecommunications Conference (GLOBECOM 2010)*. IEEE, 2010, pp. 1–6.

[3] R. Pabst, B. H. Walke, D. C. Schultz, P. Herhold, H. Yanikomeroglu, S. Mukherjee, H. Viswanathan, M. Lott, W. Zirwas, M. Dohler, H. Aghvami, D. D. Falconer, and G. P. Fettweis, "Relay-based deployment concepts for wireless and mobile broadband radio," *IEEE Communications Magazine*, vol. 42, no. 9, pp. 80–89, Sept 2004.

[4] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: a survey," *ELSEVIER Computer Networks*, vol. 47, no. 4, pp. 445–487, Mar. 2005.

[5] M. Scharf and A. Ford, "Multipath tcp (mptcp) application interface considerations," Tech. Rep., 2013.

[6] E. R. Stewart, "Stream control transmission protocol," Network Working Group at IETF, Tech. Rep., 2007.

[7] J. Iyengar, K. Shah, P. Amer, and R. Stewart, "Concurrent multipath transfer using SCTP multihoming," in *Presented at International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'04)*, San Jose, CA, July 2004.

[8] J. Iyengar, P. Amer, and R. Stewart, "Concurrent Multipath Transfer Using SCTP Multihoming Over Independent End-to-End Paths," *IEEE/ACM Transactions on Networking*, vol. 14, no. 5, pp. 951 –964, October 2006.

[9] T. Yang, L. Pan, L. Jian, H. Hongcheng, and W. Jun, "Reducing receive buffer blocking in cmt based on SCTP using retransmission policy," in *2011 IEEE 3rd International Conference on Communication Software and Networks (ICCSN)*. IEEE, 2011, pp. 122–125.

[10] J. R. Iyengar and P. D. Amer and R. Stewart, "Receive buffer blocking in concurrent multipath transfer," in *Global Telecommunications Conference, 2005. GLOBECOM'05. IEEE*, 2005.

[11] J. R. Iyengar, P. D. Amer, and R. Stewart, "Performance implications of a bounded receive buffer in concurrent multipath transfer," *Computer Communications*, vol. 30, no. 4, pp. 818–829, 2007.

[12] M. Fisk and W.-c. Feng, "Dynamic right-sizing in TCP," *http://lib-www. lanl. gov/la-pubs/00796247. pdf*, p. 2, 2001.

[13] S. Barré, C. Paasch, and O. Bonaventure, "Multipath tcp: from theory to practice," in *International Conference on Research in Networking*. Springer, 2011, pp. 444–457.

[14] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, 2011, pp. 99–112.

[15] I. A. Halepoto, F. C. M. Lau, and Z. Niu, "Scheduling over dissimilar paths using CMT-SCTP," in *Seventh International Conference on Ubiquitous and Future Networks*, July 2015, pp. 535–540.

[16] C. Xu and Z. Li and J. Li and H. Zhang and G. M. Muntean, "Cross-Layer Fairness-Driven Concurrent Multipath Video Delivery Over Heterogeneous Wireless Networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, pp. 1175–1189, 2015.

[17] P. Amer, M. Becke, T. Dreibholz, N. Ekiz, J. Iyengar, P. Natarajan, R. Stewart, and M. Tüxen, "Load sharing for the stream control transmission protocol (SCTP)," *IETF ID: draft-tuexen-tsvwg-sctp-multipath-06 (work in progress)*, 2013.

[18] M. Becke, T. Dreibholz, H. Adhari, and E.P.Rathgeb, "On the fairness of transport protocols in a multi-path environment," in *IEEE International Conference on Communications (ICC)*. IEEE, 2012, pp. 2666–2672.

[19] I. Aydin, J. Iyengar, P. Conrad, C. Shen, and P. Amer, "Evaluating TCP-friendliness in light of concurrent multipath transfer," *Computer Networks*, vol. 56, no. 7, pp. 1876 – 1892, 2012.

[20] J. Iyengar, "Concurrent multipath transfer using sctp multihoming," *Multihomed Communication with SCTP (Stream Control Transmission Protocol)*, p. 99, 2012.

[21] L. Cui, S. J. Koh, and W. J. Lee, "Fast selective ACK scheme for throughput enhancement of multi-homed SCTP hosts," *IEEE Communications letters*, vol. 14, no. 6, pp. 587–589, 2010.

[22] P. Natarajan, N. Ekiz, P. D. Amer, J. R. Iyengar, and R. Stewart, "Concurrent multipath transfer using SCTP multihoming: Introducing the potentially-failed destination state," in *International Conference on Research in Networking*. Springer, 2008, pp. 727–734.

[23] T. Dreibholz, M. Becke, H. Adhari, and E. P. Rathgeb, "On the impact of congestion control for concurrent multipath transfer on the transport layer," in *Proceedings of the 11th IEEE International Conference on Telecommunications (ConTEL)*, 2011, pp. 397–404.

[24] T. Dreibholz, M. Becke, J. Pulinthanath, and E. P. Rathgeb, "Applying TCP-friendly congestion control to concurrent multipath transfer," in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*. IEEE, 2010, pp. 312–319.

[25] C. Xu, Z. Li, J. Li, H. Zhang, and G.-M. Muntean, "Cross-layer fairness-driven concurrent multipath video delivery over heterogeneous wireless networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 7, pp. 1175–1189, 2015.

[26] N. Kuhn, E. Lochin, A. Mifdaoui, G. Sarwar, O. Mehani, and R. Boreli, "DAPS: intelligent delay-aware packet scheduling for multipath transport," in *2014 IEEE International Conference on Communications (ICC)*. IEEE, 2014, pp. 1222–1227.

[27] Y. Dong, D. Wang, N. Pissinou, and J. Wang, "Multi-path load balancing in transport layer," in *3rd EuroNGI Conference on Next Generation Internet Networks*. IEEE, 2007, pp. 135–142.

[28] D. S. Lun, M. Médard, R. Koetter, and M. Effros, "Further results on coding for reliable communication over packet networks," in *Proceedings. International Symposium on Information Theory (ISIT 2005)*. IEEE, 2005, pp. 1848–1852.

[29] ——, "On coding for reliable communication over packet networks," *Physical Communication*, vol. 1, no. 1, pp. 3–20, 2008.

[30] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[31] B. Francis, V. Narasimhan, A. Nayak, and I. Stojmenovic, "Techniques for enhancing tcp performance in wireless networks," in *32nd International Conference on Distributed Computing Systems Workshops*. IEEE, 2012, pp. 222–230.

[32] G. Jakllari, S. Eidenbenz, N. Hengartner, S. Krishnamurthy, and M. Faloutsos, "Link positions matter: A noncommutative routing metric for wireless mesh networks," *IEEE Transactions on Mobile Computing*, vol. 11, no. 1, pp. 61–72, Jan 2012.

[33] D. J. Leith, Q. Cao, and V. G. Subramanian, "Max-min Fairness in 802.11 Mesh Networks," *IEEE/ACM Transactions on Networking*, vol. 20, no. 3, pp. 756–769, Jun. 2012.

[34] X. M. Zhang, W. B. Zhu, N. N. Li, and D. K. Sung, "TCP congestion window adaptation through contention detection in ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 9, pp. 4578–4588, 2010.

[35] J. Karlsson, A. Kassler, and A. Brunstrom, "Impact of packet aggregation on TCP performance in wireless mesh networks," in *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks Workshops, (WoWMoM 2009)*, June 2009, pp. 1–7.

[36] R. De Oliveira and T. Braun, "A smart TCP acknowledgment approach for multihop wireless networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 2, pp. 192–205, 2007.

[37] Z. Fu, H. Luo, P. Zerfos, S. Lu, L. Zhang, and M. Gerla, "The impact of multihop wireless channel on TCP performance," *IEEE transactions on Mobile Computing*, vol. 4, no. 2, pp. 209–221, 2005.

[38] K. Nahm, A. Helmy, and C.-C. Jay Kuo, "TCP over multihop 802.11 networks: issues and performance enhancement," in *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, ser. MobiHoc '05. ACM, 2005, pp. 277–287.

[39] Y. Su, P. Steenkiste, and T. Gross, "Performance of TCP in Multi-Hop Access Networks," in *16th International Workshop on Quality of Service (IWQoS 2008)*, 2008, pp. 181–190.

[40] T. Kuang and C. Williamson, "A bidirectional multi-channel mac protocol for improving TCP performance on multihop wireless ad hoc networks," in *Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ser. MSWiM '04. New York, NY, USA: ACM, 2004, pp. 301–310.

[41] S. Xu and T. Saadawi, "Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks?" *IEEE Communications Magazine*, vol. 39, no. 6, pp. 130–137, Jun 2001.

[42] E. Altman and T. Jiménez, "Novel delayed ACK techniques for improving TCP performance in multihop wireless networks," in *8th International Conference on Personal Wireless Communications (PWC 2003)*, vol. 2775. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 237–250.

[43] K. Xu and S. Bae and S. Lee and M. Gerla, "TCP behavior across multihop wireless networks and the wired internet," in *Proceedings of the 5th ACM international workshop on Wireless mobile multimedia*. ACM, 2002, pp. 41–48.

[44] N. Arianpoo, P. Jokar, and V. C. Leung, "Enhancing TCP performance in wireless mesh networks by cross layer design," in *International Conference on Computing, Networking and Communications (ICNC 2012)*, 2012.

[45] N. Arianpoo and V. C. Leung, "How network monitoring and reinforcement learning can improve tcp fairness in wireless multi-hop networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2016, no. 1, p. 278, 2016.

[46] A. Al-Jubari, M. Othman, B. Mohd Ali, and N. Abdul Hamid, "An Adaptive Delayed Acknowledgment Strategy to Improve TCP Performance in Multi-hop Wireless Networks," *Wireless Personal Communications*, vol. 69, no. 1, pp. 307–333, 2013.

[47] T. Yuki, T. Yamamoto, M. Sugano, M. Murata, H. Miyahara, and T. Hatauchi, "Performance improvement of TCP over an ad hoc network by combining of data and ACK packets," *The Institute of Electronics, Information and Communication Engineers (IEICE) Transactions on Communications*, vol. 86, pp. 3559–3568, 2004.

[48] K. Xu, M. Gerla, L. Qi, and Y. Shu, "Enhancing TCP fairness in ad hoc wireless networks using neighborhood RED," in *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking (ACM MobiCom 2003)*. New York, NY, USA: ACM, 2003, pp. 16–28.

[49] J. Ye, J.-X. Wang, and J.-W. Huang, "A cross-layer TCP for providing fairness in wireless mesh networks ," *International Journal of Communication Systems*, vol. 24, no. 12, pp. 1611–1626, 2011.

[50] A. Raniwala, D. Pradipta, and S. Sharma, "End-to-End Flow Fairness Over IEEE 802.11-Based Wireless Mesh Networks," in *26th IEEE International Conference on Computer Communications (INFOCOM 2007)*, May 2007, pp. 2361–2365.

[51] S. ElRakabawy and C. Lindemann, "A Practical Adaptive Pacing Scheme for TCP in Multihop Wireless Networks," *IEEE/ACM Transactions on Networking*, vol. 19, no. 4, pp. 975–988, 2011.

[52] H. Xie, A. Boukerche, and A. Loureiro, "TCP-ETX: A cross layer path metric for TCP optimization in wireless networks," in *IEEE International Conference on Communications (ICC 2013)*, June 2013, pp. 3597–3601.

[53] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, pp. 89–102, 2002.

[54] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-aware datacenter TCP (D2TCP)," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, 2012.

[55] V. Gambiroza, B. Sadeghi, and E. W. Knightly, "End-to-end performance and fairness in multihop wireless backhaul networks," in *Proceedings of the 10th annual international conference on Mobile computing and networking*, ser. MobiCom '04. New York, NY, USA: ACM, 2004, pp. 287–301.

[56] S. Shioda, H. Iijima, T. Nakamura, S. Sakata, Y. Hirano, and T. Murase, "ACK pushout to achieve TCP fairness under the existence of bandwidth asymmetry," in *Proceedings of the 5th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, ser. PM2HW2N '10, 2010, pp. 39–47.

[57] C. Cicconetti, I. Akyildiz, and L. Lenzini, "FEBA: A Bandwidth Allocation Algorithm for Service Differentiation in IEEE 802.16 Mesh Networks," *IEEE Transactions on Networking*, vol. 17, no. 3, pp. 884–897, 2009.

[58] ——, "Bandwidth Balancing in Multi-Channel IEEE 802.16 Wireless Mesh Networks," in *Proceedings of 26th IEEE International Conference on Computer Communications (ICC 2007)*, may 2007, pp. 2108 –2116.

[59] T.-C. Hou, C.-W. Hsu, and C.-S. Wu, "A Delay-based Transport Layer Mechanism for Fair TCP Throughput over 802.11 Multihop Wireless Mesh Networks ," *International Journal of Communication Systems*, vol. 24, no. 8, pp. 1015–1032, August 2011.

[60] S. Fowler, M. Eberhard, and K. Blow, "Implementing an adaptive TCP fairness while exploiting 802.11e over wireless mesh networks," *International Journal of Pervasive Computing and Communications*, vol. 5, pp. 272 – 294, 2009.

[61] T. Li, D. J. Leith, V. Badarla, D. Malone, and Q. Cao, "Achieving End-to-end Fairness in 802.11e Based Wireless Multi-Hop Mesh Networks Without Coordination," *Mobile Networks and Applications*, vol. 16, no. 1, pp. 17–34, Feb. 2011.

[62] K. Xu and N. Ansari, "Stability and fairness of rate estimation-based AIAD congestion control in TCP," *IEEE Communications Letters*, vol. 9, no. 4, pp. 378–380, April 2005.

[63] K. L. E. Law and W.-C. Hung, "Engineering TCP transmission and retransmission mechanisms for wireless networks." *Pervasive and Mobile Computing*, vol. 7, no. 5, pp. 627–639, 2011.

[64] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control (BIC) for fast long-distance networks," in *23rd IEEE Conference on Computer and Communications Societies (INFOCOM 2004)*, vol. 4, 2004, pp. 2514–2524.

[65] S. Ha, I. Rhee, and L. Xu, "Cubic: a new tcp-friendly high-speed tcp variant," *ACM Special Interest Group on Operating Systems Review (ACM SIGOPS 2008)*, vol. 42, no. 5, pp. 64–74, 2008.

[66] C. P. Fu and S. Liew, "TCP Veno: TCP enhancement for transmission over wireless access networks," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 2, pp. 216–228, February 2003.

[67] K. Winstein, A. Sivaraman, and H. Balakrishnan, "Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks," in *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, 2013, pp. 459–471.

[68] K. Winstein and H. Balakrishnan, "TCP ex machina: computer-generated congestion control," in *ACM Special Interest Group on Data Communication conference 2013 (SIGCOMM 2013)*, 2013, pp. 123–134.

[69] K. Nichols and V. Jacobson, "Controlling queue delay," *Communications of the ACM*, vol. 55, no. 7, pp. 42–50, 2012.

[70] K. Xu, Y. Tian, and N. Ansari, "Tcp-jersey for wireless ip communications," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 4, pp. 747–756, May 2004.

[71] I. Aydin and C. Shen, "Performance Evaluation of Concurrent Multipath Transfer Using SCTP Multihoming in Multihop Wireless Networks," in *8th IEEE International Symposium on Network Computing and Applications (NCA 2009)* , July 2009, pp. 234–241.

[72] Dreibholz, T. and Becke, M. and Pulinthanath, J. and Rathgeb, E.P., "Applying TCP-Friendly Congestion Control to Concurrent Multipath Transfer," in *24th IEEE International Conference on Advanced Information Networking and Applications (AINA 2010)* , April 2010, pp. 312–319.

[73] Y. Cao and Ch. Xu and J. Guan and H. Zhang, "TCP-friendly CMT-based multimedia distribution over multi-homed wireless networks," in *IEEE Wireless Communications and Networking Conference (WCNC 2014)* , April 2014, pp. 3028–3033.

[74] I. Aydin, "SCTP-based Concurrent Multipath Transfer in the Contexts of Multihop Wireless Networks and Tcp-friendliness," Ph.D. dissertation, University of Delaware, 2010.

[75] C. Xu and T. Liu and J. Guan and H. Zhang and G. M. Muntean, "CMT-QA: Quality-Aware Adaptive Concurrent Multipath Data Transfer in Heterogeneous Wireless Networks," *IEEE Transactions on Mobile Computing*, vol. 12, pp. 2193–2205, 2013.

[76] Y. Cao and Ch. Xu and J. Guan and F. Song and H. Zhang, "Environment-aware CMT for efficient video delivery in wireless multimedia sensor networks," *International Journal of Distributed Sensor Networks*, pp. 4522–4527, 2012.

[77] Y. Cao and Ch. Xu and J. Guan and J. Zhao and H. Zhang, "Cross-layer cognitive CMT for efficient multimedia distribution over multi-homed wireless networks," *IEEE Wireless Communications and Networking Conference (WCNC 2013)* , pp. 4522–4527, April 2013.

[78] P. Natarajan, N. Ekiz, P. D. Amer, J. Iyengar, and R. Stewart, "Concurrent multipath transfer using sctp multihoming: Introducing the potentially-failed destination state," in *Proceedings of 7th International IFIP-TC6 Networking Conference Singapore*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 727–734.

[79] W. Zhuang and N. Mohammadizadeh and X. Shen, "Multipath transmission for wireless Internet access–from an end-to-end transport layer perspective," *Journal of Internet Technology*, vol. 13, pp. 1–18, 2012.

[80] Becke, M. and Dreibholz, T. and Bayer, A. and Packeiser, M. and Rathgeb, E.P., "Alternative transmission strategies for multipath transport of multimedia streams over wireless networks," in *12th International Conference on Telecommunications (ConTEL 2013)* , June 2013, pp. 147–154.

[81] Y. Yuan and Z. Zhang and J. Li and J. Shi and J. Zhou and G. Fang and E. Dutkiewicz, "Extension of SCTP for Concurrent Multi-Path Transfer with Parallel Subflows," in *IEEE Wireless Communications and Networking Conference (WCNC 2010)* , April 2010, pp. 1–6.

[82] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on information theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[83] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading structure for randomness in wireless opportunistic routing," *ACM SIGCOMM Computer Communication Review*, vol. 37, pp. 169–180, October 2007.

[84] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "Xors in the air: practical wireless network coding," in *ACM SIGCOMM computer communication review*, vol. 36, no. 4. ACM, 2006, pp. 243–254.

[85] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 5, pp. 782–795, 2003.

[86] J. Sundararajan, D. Shah, M. Medard, M. Mitzenmacher, and J. Barros, "Network Coding Meets TCP," in *Proc. IEEE INFOCOM*, Toronto, Canada, April 2009, pp. 280 –288.

[87] J. K. Sundararajan, S. Jakubczak, M. Medard, M. Mitzenmacher, and J. Barros, "Network Coding Meets TCP: Theory and Implementation," *Proceedings of the IEEE*, vol. 99, pp. 490 – 512, March 2011.

[88] C. J.C.H. Watkins and P. Dayan, "Technical Note: Q-Learning," *Machine Learning*, 1992.

[89] C. J. Watkins and P. Dayan, "Technical note: Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[90] E. Even-Dar and Y. Mansour, "Learning rates for q-learning," *Journal of Machine Learning Research*, vol. 5, pp. 1–25, Dec. 2004.

[91] L. Matignon, G. J. Laurent, and N. Le Fort-Piat, "Reward function and initial values: better choices for accelerated goal-directed reinforcement learning," in *International Conference on Artificial Neural Networks*. Springer, 2006, pp. 840–849.

[92] M. Kearns and S. Singh, "Finite-sample convergence rates for q-learning and indirect algorithms," in *In Neural Information Processing Systems 12*. MIT Press, 1999, pp. 1996 – 1002.

[93] L. Matignon and G.J. Laurent and N. Le Fort-Piat, "Improving Reinforcement Learning Speed for Robot Control," *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 3172–3177, 2006.

[94] RaspberryPi, "Model B," (Accessed on 2017-02-20). [Online]. Available: https://www.raspberrypi.org/

[95] "The FreeBSD Project," (Accessed on 2017-02-20). [Online]. Available: https://www.freebsd.org/

[96] "Debian Wheezy," (Accessed on 2017-02-20). [Online]. Available: https://www.debian.org/releases/wheezy/

[97] TPLINK, "150mbps wireless n nano usb adapter," (Accessed on 2017-02-20). [Online]. Available: http://www.tp-link.com/lk/products

[98] Wireshark. (Accessed on 2017-02-20). [Online]. Available: https://www.wireshark.org

[99] G. King and L. Zeng, "Logistic regression in rare events data," *Political Analysis*, vol. 9, pp. 137–163, 2001.

[100] J. K. Sundararajan, D. Shah, and M. Médard, "ARQ for network coding," in *2008 IEEE International Symposium on Information Theory*. IEEE, 2008, pp. 1651–1655.

[101] G. Bolch and S. Greiner and H. de Meer and K. S. Trivedi, *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*. Wiley-Interscience, 1998.

[102] Han, Yijie and Makowski, Armand M, "Resequencing delays under multipath routing–Asymptotics in a simple queueing model," Institute for System Research, University of Maryland, Tech. Rep., 2005.

[103] "QualNet," (Accessed on 2017-02-20). [Online]. Available: http://web.scalable-networks.com/qualnet-network-simulator-software

[104] B. Horan, *Practical Raspberry Pi*. Apress, 2013.

[105] J. Sundararajan, D. Shah, M. Medard, and P. Sadeghi, "Feedback-based online network coding," Nov. 29 2011, US Patent 8,068,426.

[106] J. Sundararajan, D. Shah, and M. Medard, "Online network coding for optimal throughput and delay - the three-receiver case," in *International Symposium on Information Theory and Its Applications (ISITA 2008)*, Dec 2008, pp. 1–6.

[107] J. Barros, R. Costa, D. Munaretto, and J. Widmer, "Effective delay control in online network coding," in *INFOCOM 2009, IEEE*, April 2009, pp. 208–216.

[108] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz, "A comparison of mechanisms for improving tcp performance over wireless links," *IEEE/ACM Transactions on Networking*, 1997.

[109] S. Kopparty, S. Krishnamurthy, M. Faloutsos, and S. Tripathi, "Split TCP for mobile ad hoc networks," in *IEEE Global Telecommunications Conference (GLOBECOM '02)*, vol. 1, November 2002, pp. 138–142.

[110] M. Gerla and K. Tang and R. Bagrodia, "TCP performance in wireless multi-hop networks," in *IEEE 2nd Workshop on Mobile Computing Systems and Applications (WMCSA 99)* , 1999, pp. 41–50.

[111] R. Jain, D.-M. Chiu, and W. R. Hawe, *A quantitative measure of fairness and discrimination for resource allocation in shared computer system*. Eastern Research Laboratory, Digital Equipment Corporation Hudson, MA, 1984, vol. 38.

[112] N. Jani and K. Kant, "SCTP Performance in Data Center Environments," in *Proceedings of the 2005 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'05)*, 2005.

[113] P. Domingos and M. Pazzani, "On the optimality of the simple bayesian classifier under zero-one loss," *Machine Learning*, vol. 29, pp. 103–130, 1997.

[114] M. Franceschinis, M. Mellia, M. Meo, and M. Munafo, "Measuring TCP over WiFi: A real case," in *1st workshop on Wireless Network Measurements (Winmee), Riva Del Garda, Italy*, 2005.

[115] V. Jacobson and R. Braden and D. Borman and M. Satyanarayanan and JJ Kistler and LB Mummert and MR Ebling, "RFC 1323: TCP extensions for high performance," May 1999.

[116] N. Brownlee and K. C. Claffy, "Understanding internet traffic streams: dragonflies and tortoises," *IEEE Communications Magazine*, vol. 40, no. 10, pp. 110–117, Oct 2002.

[117] Z. Fu, H. Luo, P. Zerfos, S. Lu, L. Zhang, and M. Gerla, "The impact of multihop wireless channel on TCP performance," *IEEE Transactions on Mobile Computing*, vol. 4, no. 2, pp. 209–221, March 2005.