Cross-layer wireless networking emulation in EMANE

by

Zehua Li

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Jia Liu, Major Professor
Ying Cai
Sheng Bao

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this thesis. The Graduate College will ensure this thesis is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2019

TABLE OF CONTENTS

Page

LIST OF FIGURES iii

NOMENCLATURE iv

ACKNOWLEDGMENTS v

ABSTRACT vi

CHAPTER 1. INTRODUCTION: QUEUE-LENGTH BASED ALGORITHMS 1
1.1 Networking History 1
1.2 System Model 2

CHAPTER 2. SINGLE-HOP IMPLEMENTATION 5
2.1 Pathloss Holder 5
2.2 MaxWeight Scheduling 7
2.3 Congestion Control 8
2.3.1 Changing the Source Rate 9

CHAPTER 3. MULTI-HOP IMPLEMENTATION 10
3.1 Queue Length Differential 11
3.2 Centralized Scheduler 11

CHAPTER 4. RESULT 13
4.1 Experiment Setup 13
4.2 Experiment Evaluation 13

CHAPTER 5. CONCLUSION 15
5.1 Overview of Contributions 15
5.2 Future Work 15

REFERENCES 16

LIST OF FIGURES

Page

Figure 1.1 Single-hop model setup 3

Figure 1.2 Simulation result in MATLAB 4

Figure 2.1 Single-hop EMANE with new components 6

Figure 3.1 Multi-Hop model setup 10

Figure 4.1 Source rate vs time 14

Figure 4.2 Queue length vs time 14

NOMENCLATURE

QLA	Queue-Length-based Algorithm
CSI	Channel State Information
OSI	Open Systems Interconnection
GMM	Greedy Maximal Matching
MGEN	Multi-Generator
EMANE	Extendable Mobile Ad-hoc Network Emulator

ACKNOWLEDGMENTS

I would like to thank my committee chair, Jia Liu who gave me great support during my Master study. I would also like to thank my committee members, Ying Cai and Sheng Bao for their guidance and support throughout the course of this research. In addition, I would also like to thank my colleagues, the department faculty and staff for making my time at Iowa State University a wonderful experience.

I would like to dedicate this thesis to my wife Baoyue, without whose support I would not have been able to complete this work so smoothly. I would like to thank my family and friends for their loving guidance and financial assistance during the writing of this work.

ABSTRACT

After MaxWeight scheduling algorithm came out, Cross-layer wireless networking optimization became a popular direction on enhancing wireless networking performance to the next level. To support research in Cross-layer wireless networking optimization, I developed a set of tools: Pathloss holder, MaxWeight scheduler, Dynamic Multi-Generator (D-MGEN) and Centralized scheduler based on Extendable Mobile Ad-hoc Network Emulator (EMANE) [1]. Using the four major components, I built the connection between existing protocols. Contribute to it, researchers now can test their optimization algorithms, including first-order momentum-based algorithms and second-order algorithms in a more realistic environment.

CHAPTER 1. INTRODUCTION: QUEUE-LENGTH BASED ALGORITHMS

Nowadays, wireless networking becomes more and more important in our daily life. Topics like 5G, IoT and swarm networking are getting hotter than ever. Researchers who had joined in this topic and put effort on improving networking performance found that there was a big block on the way: wireless channel resources are limited in practice. Therefore, how to utilize the existing channel resources in an efficient way and how to improve the throughput in wireless networking become a very attracting challenge.

1.1 Networking History

Since ISO and CCITT published the OSI model, [2] i.e. standard ISO 7498 and standard X.200 in the year 1984, people began to follow it and design different internet protocols that work in their own areas. According to this standard OSI reference, one can separate the intricate networking process into simpler components (protocols). Contribute to it, developers can not only define behaviors in a single protocol while keeping other components unchanged, but also combine multiple protocols to fit different scenarios. Not surprisingly, these advantages have greatly boosted the growth of the internet empire in recent years.

When applying OSI model to wireless networking, people can support wireless hardware by changing only the MAC layer and PHY layer in OSI model. Then migrate all existing wired networking technologies to the wireless networking. It is a great starting point where users can enjoy the WIFI/cellular technology in an unconscious way. However, followed by the rapid growth of wireless networking, problems that have no effect in traditional wired networking become conspicuous into our view. For instance, the electromagnetic radiation which wireless networking is built on has a limited bandwidth,

with more and more mobile devices join in the army, the bandwidth allocation is tightly stretched when supporting a large number of devices simultaneously. This problem encourages people to think of a better way of scheduling the packets and find how we can transmit packets under the limited channel resources effectively.

These years, people have developed different methods to schedule packets under the restricted bandwidth such as FDMA [3], TDMA [4] and CDMA [4]. But not a single one requires information from other protocols, which can lead to an uncertainty of achieving optimal throughput.

1.2 System Model

In the year 1992, the queue-length-based algorithm with the cross-layer concept of optimization [5] was first proposed by Leandros Tassiulas in his publication *"Jointly optimal routing and scheduling in packet ratio networks"*. Single-hop networking, as shown in Figure 1.1, has a base station which can serve N users, and a time slot-based system with time slot parameter $t \in \{0,1,2, \dots\}$. As time elapsing, the channel state $\pi[t]$ will change erratically due to the indeterministic of noise and interference. Now define $q_n[t]$ as the queue length parameter. After receiving the data, the base station will hold data packets in queues for each user, i.e. $q_1[t], q_2[t], \dots, q_N[t]$. Then based on $\pi[t]$ and $q_n[t]$, the algorithm can select service rates $s_1[t], \dots, s_N[t]$ from the feasible service rate region $\mathcal{C}_{\pi[t]} \in \mathbb{R}_+^N$ at each channel state.

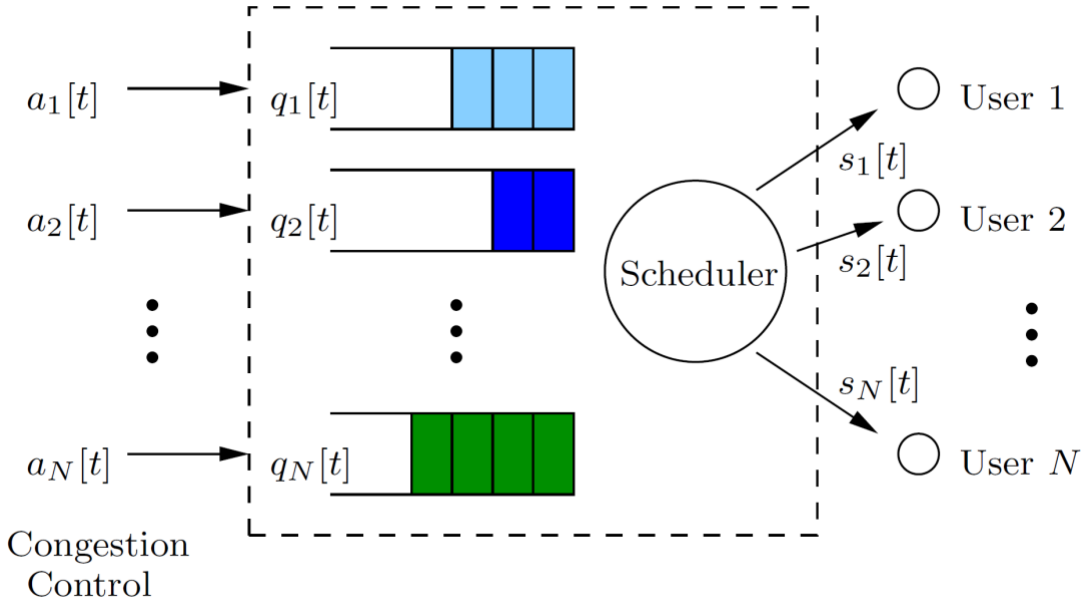

Figure 1.1 Single-hop model setup

In the year 2006, Lin used Lagrangian dual and gradient algorithm to solve this QLA problem [6]. Given queue length and channel state information $q[t]$, $\pi[t]$, one can solve the MaxWeight scheduling

$$\boldsymbol{s}[t] = arg \max_{x \in C_{\pi[t]}} (\sum_{n=1}^{N} q_n[t] x_n) \tag{1}$$

to get the serving rate in each time slot. Then determine the arrival rate $a_n[t]$ by solving function

$$a_n[t] = arg \max_{a \in [0, a^{max}]} \{KU_n(a) - q_n[t]a\} \tag{2}$$

And finally, update queue length as follows in each time slot:

$$q_n[t+1] = (q[t] - s_n[t] + a_n[t])^+ \tag{3}$$

Inspired by Lin, Jia Liu further enhances the convergence rate by involving a momentum algorithm called Heavyball [7] into the problem. Under the same Single-hop model, he redefines the MaxWeight scheduler as follows:

$$\boldsymbol{s}[t] = arg \max_{x \in \mathcal{C}_{\pi[t]}} \left(\sum_{n=1}^{N} \boldsymbol{w}_n[t] x_n\right) \tag{4}$$

Compare to the original scheduler function, Liu introduces a new weight variable $\boldsymbol{w}[t]$ as following

$$\boldsymbol{w}_n[t+1] = \left(\boldsymbol{w}_n[t] + \Delta \boldsymbol{q}_n[t] + \beta(\boldsymbol{w}_n[t] - \boldsymbol{w}_n[t-1])\right)^+ \tag{6}$$

to replace the queue length $\boldsymbol{q}[t]$ when scheduling the serving packets. It is clear that $\boldsymbol{w}[t]$ depends on $\boldsymbol{q}[t]$, where $\beta \in [0,\ 1)$ is a system parameter that controls the weight of momentum term in Heavyball algorithm. Apply the same idea to congestion control he gets

$$a_n[t] = arg \max_{a \in [0, a^{max}]} \{KU_n(a) - \boldsymbol{w}_n[t]a\} \tag{5}$$

Turns out the adjusted formulas will not influence how queue length is being updated. And Liu successfully optimized the throughput without losing any performance. The new approach also achieved a better estimation by adding memory term into the calculation.

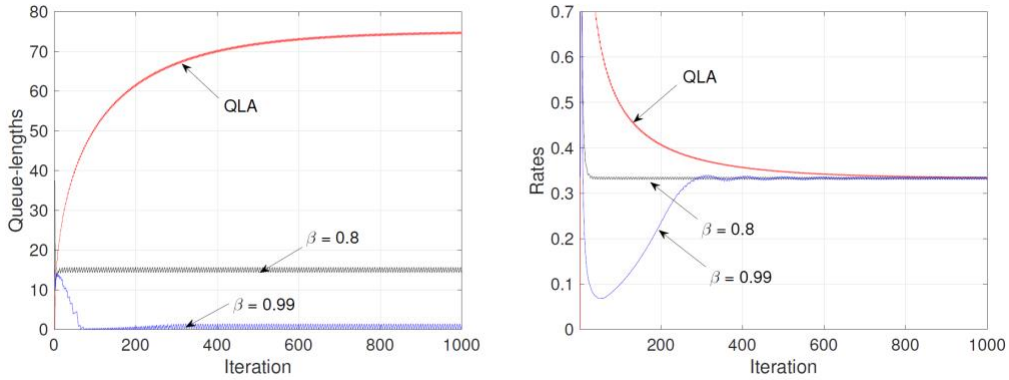

Figure 1.2 Simulation result in MATLAB

CHAPTER 2. SINGLE-HOP IMPLEMENTATION

My goal is to construct a program that can help with implementing the cross-layer optimization algorithm. To achieve it, I developed three major components: Pathloss holder, MaxWeight scheduler and Dynamic MGEN (See Figure 2.1). By Combining these components, I successfully integrated the traditional OSI model into cross-layer networking.

2.1 Pathloss Holder

In wireless networking, it is very important for MAC protocols to find an efficient way of occupying the channel. Engineers in the past have different ways to arrange the channel resources. Traditional protocols like FDMA and TDMA will carefully assign the channel resources to different users without knowing the current channel state information (CSI). However, the ignorant of noise and interference will lead to a data lost during the transmission process. So, obtaining the noises in real-time slot is the key point to improve throughput in wireless networking. Unfortunately, knowing the noise is not an easy job. Researchers found that noises and interferences always occur in various reasons, and the strength of noises is shaking all the time.

When getting CSI in each time slot, I noticed that a part of implementation in EMANE platform named ‘emaneeventservice’ will take a predefined data (scenario file) as input, and broadcast events to all NEMs. Network Emulation Modules (NEMs) is a network stacks provide by EMANE, which majorly consist of a MAC layer and a PHY layer. During the emulation process, PHY protocol in EMANE will receive the events, keep real-time CSI and emulate the transmission process. My target now is to allow MAC layer also to be able to get the CSI from PHY layer.

My strategy is to design a global static variable named ‘pathlossholder’ to store the real-time CSI. Each time when PHY protocol receives CSI from ‘emaneeventservice’, the ‘pathlossholder’ value will update automatically. With ‘pathlossholder’ being static, the current processes will not be blocked when PHY protocols and MAC protocols exchange their CSI data. Benefit from the Pathloss holder, I can connect MAC layer to PHY layer, as well as help MAC layer uses the MaxWeight algorithm for scheduling.

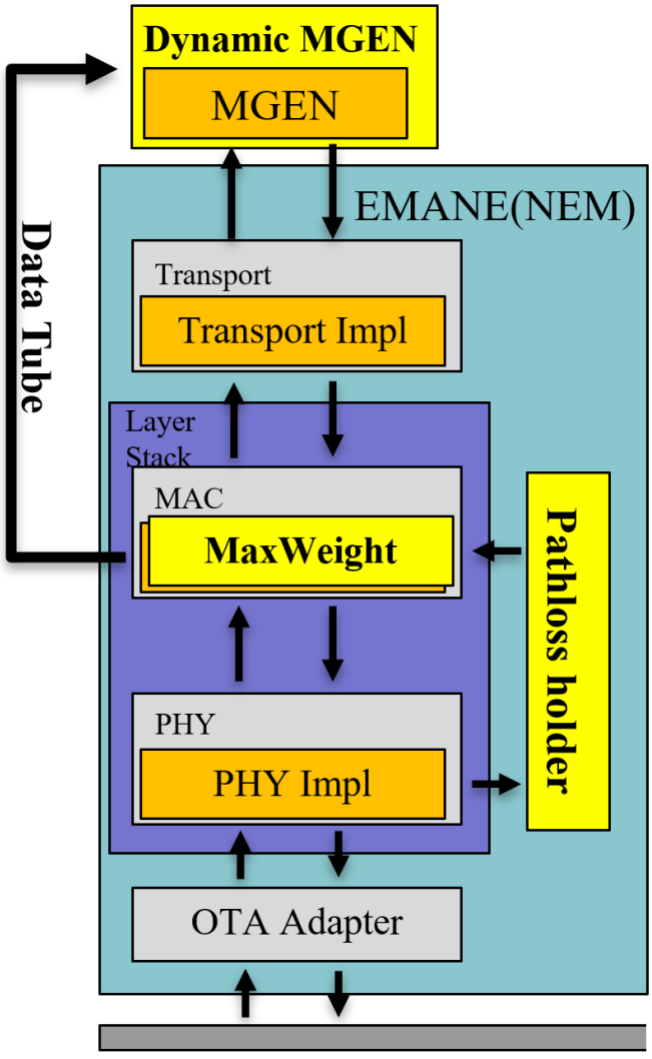

Figure 2.1 Single-hop EMANE with new components

2.2 MaxWeight Scheduling

When coming to MAC protocol, developers will primarily consider two questions, i.e. which destination to serve and how to occupy channel resources at the current time slot. There are many good protocols existing for solving these problems such as CDMA, FDMA, and TDMA. For this project, I built the MaxWeight algorithm in EMANE based on the TDMA protocol.

In traditional TDMA, people must assign a node to each time slot. It will bring a large chance on time wasting if the current user node is idle. For efficiency purpose, I decided to change the behavior of TDMA in EMANE so it can dynamically select which user node to serve. Firstly, I split each time frame into two time slots. Next, I ask all user nodes to listen from the host node at the first time slot, then opposite the listening directions at the second time slot. Finally, I let the host node decide which user to serve based on a given scheduling algorithm.

After changing the behavior, I started to implement the MaxWeight algorithm in TDMA module. I read queue length from TDMA and CSI from 'pathlossholder'. By solving equation (4), MaxWeight algorithm can decide which destination to serve in the current time slot. At this moment, the host node serves only one destination per time slot.

As a momentum approach for MaxWeight, Heavyball algorithm needs n (n is the number of users in the model) spaces to store weights from the last k iterations. So, I introduced a private list variable with length k to keep k steps in weight history. This helps researchers specify the length of the historical information they needed for scheduling.

2.3 Congestion Control

Congestion control is one of the major functionalities in TCP protocol which can avoid packet loss and reduce delay time during packet transmission. After Van Jacobson published his paper *"Congestion Avoidance and Control"* in 1988 [8], people start to agree and use a standard interface for designing the TCP congestion control method. This interface can take acknowledgments through each transmission and apply it to adjust the window size. There are many different algorithms that can improve the performance of congestion control and offer great performance in wired networking.

In wireless networking, knowing only the acknowledgment information is not enough to help control the window size. It can easily lead to a failure in achieving optimal performance. In cross-layer optimization design, people are working on involving queue length from MAC layer to help adjust the congestion control window size, since it can bring multiple advantages on improving performance. First, compare to acknowledgment, it is easier to obtain queue length only from the local machine. Acknowledgment was used to transmit via cable in wired networking. It is costless in the sense that cables can afford gigantic number of bits. But in wireless networking, the only way to transmit acknowledgements is using channel resources, and it is quite expensive since wireless networking has limited bandwidth. In contrary, queue length is much cheaper because it stays in the local machine. Moreover, only using acknowledgment is time consuming as the transmitter has to wait for receiver to send information back. Most importantly, the queue length information can be obtained instantly and respond to congestion control almost immediately. This feature will enormously help speed up the window size changing rate.

2.3.1 Changing the Source Rate

In general, EMANE is a platform that emulates the MAC layer and PHY layer protocols. However, it does not contain any transport protocols. Instead, EMANE can work with any standard upper-level layer protocols like TCP/IP by using the TUN/TAP.

Because of the lack of emulated transport protocol, I cannot implement congestion control algorithm in the EMANE software. And it is difficult to adjust the window size in TCP protocol since TCP runs in kernel space. In order to successfully emulate QLA algorithms, I decided to control the source generating rate accordingly based on queue length information gained from EMANE. Then, I used UDP protocol to eliminate the unwanted congestion control from TCP.

There are many different popular tools for generating packets. I selected the Multi-Generator (MGEN) [9] which is developed by the Naval Research Laboratory (NRL). MGEN has a python API that can allow users to control the network flows in real-time with simple python scripts. By taking this advantage, I developed the source rate control program in python, and named it D-MGEN. D-MGEN uses the named pipe to let the EMANE MAC protocol send queue length information to it in each time slot. More specifically, D-MGEN first initializes a named pipe, then begins to listen to the queue length information when it starts to control the flow. When EMANE starts to read packets from TUN/TAP, the MAC protocol will write the queue length information through the named pipe immediately.

CHAPTER 3. MULTI-HOP IMPLEMENTATION

In some scenario, Multi-hop networking plays a more important role than Single-hop networking dose. In a wireless networking environment, each node needs to communicate with all other nodes in the most efficient fashion (See figure 3.1). Unlike single-hop model, Multi-hop networking can guarantee the whole system work properly when any node died. However, it is also more challenging to implement cross-layer optimization algorithms into Multi-hop networking.

In Multi-hop model, the QLA algorithm falls into backpressure algorithm, and there are two major differences compare with Single-hop networking. Firstly, instead of using queue length directly in MaxWeight, backpressure using queue length differential between neighbor nodes to calculate the MaxWeight function. Secondly, since all the nodes can transmit data in Multi-hop model, it is important to have a scheduler decide which node should occupy the channel resources during the time slot (See Figure 3.1).

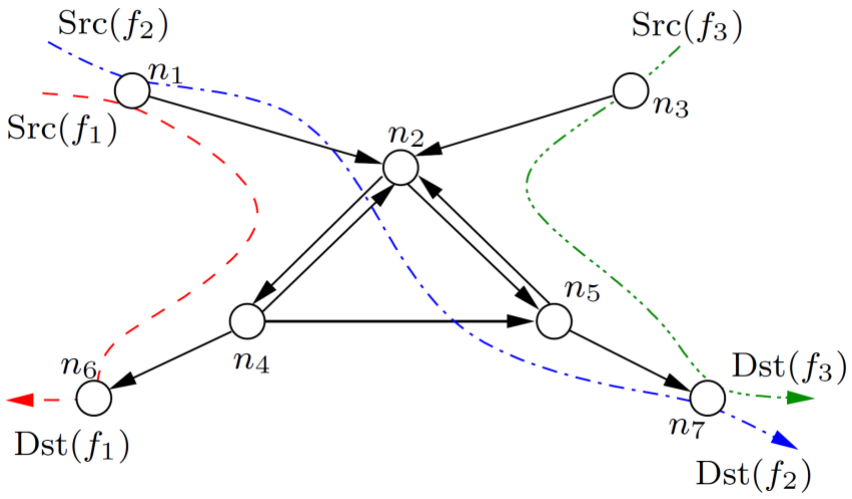

Figure 3.1 Multi-Hop model setup

3.1 Queue Length Differential

As stated in previous content, backpressure algorithm uses queue length differentials between neighbors to schedule network flows in Multi-hop emulation environment. Queue length differentials are calculated by subtracting the current node queue length from its neighbors' queue length. To find queue length from neighbors, one can either allocate a new channel or add overhead cost to current channel for nodes to transmit queue length information. Theoretically, allocating new channel will have less effect on the current data channel. However, allocating a different channel in real-world situation is usually very expensive. Besides, people cannot fully utilize this separated channel at any time, so it will lead to a waste of channel resources. In fact, this additional channel can only transmit very little information at the beginning of each time slot. At first glance, it seems like the latter approach will need more time from the channel to transmit queue length data. In fact, as long as we keep the queue length information in a comparably small size, it can save more resources. Therefore, I decided to choose the second approach for my implementation.

In details, I first defined a new packet type 'QUEUELENGTH' in EMANE. At the beginning of each time slot, the program in MAC layer will generate a message that encodes its queue length information into a small packet with type 'QUEUELENGTH'. Then, this message will be transmitted broadcast to all its neighbors. After EMANE receives the 'QUEUELENGTH' message, it will start to calculate queue length differentials. Each node must calculate the queue length differentials for all its neighbors to find the maximum queue length differential. Then, I need a scheduler to active the nodes in each time slot.

3.2 Centralized Scheduler

It is common knowledge that avoids channel interference is very important in wireless networking. To prevent interference in this project, I used the following

assumptions. First, I apply the node exclusive model [10] to emulate the real-world interference scenario. Then I use the out of band channels to communicate centralized scheduler with EMANE nodes. Finally, I engage the Greedy Maximal Matching (GMM) [10] to solve scheduling problem. Based on these, I designed my centralized scheduler in a hierarchical and modularized way, so it can support different interference models and scheduling algorithms easily. Moreover, the centralized scheduler is written in C++ to ensure performance.

The workflow of the centralized scheduler is the following:

1. At the beginning of each time slot, all nodes broadcast their queue-length information to each of their neighbors.
2. Base on the neighbors' queue length information, calculating the max queue length differentials for each link.
3. All EMANE nodes transmit its max queue length differentials to the centralized scheduler.
4. After receiving all queue length differentials from EMANE nodes, centralized scheduler using a given algorithm and interference model to decide which node need to be activated.
5. Centralized scheduler transmits the decision back to EMANE nodes, and EMANE starts to transmit.

CHAPTER 4. RESULT

To test the project, I set up a three-client Single-hop model. Based on the theoretical result, I make up the same wireless networking scenario to get a better estimation. My desired goal is to observe a similar trend like theoretical outcomes.

4.1 Experiment Setup

During the experiment, I set the channel bandwidth to 1Mhz, the channel frequency to 2.4Ghz, the transmitting power to 0db and the time slot duration to 1.5ms. Then I assign two time slots in each time frame to simplify the configuration for TDMA scheduler. For the configuration of source generator MGEN, I adjust the packet size to 512kb, and the max packet generation rate to 100 per second. Based on the above settings, I run my test 4 times by changing the momentum factor β to 0, 0.2, 0.5 and 0.8 accordingly.

4.2 Experiment Evaluation

After running the emulation, I obtained a similar result compared to the MATLAB simulation outcome. When $\beta = 0$, the Heavyball algorithm falls back to MaxWeight algorithm, and takes approximately 0.1 seconds to converge. The averaged total queue length in host node is nearly 35 packets. When setting β to 0.5, the convergence speed becomes much faster. It takes only 0.02 seconds to converge to the optimal throughput. And the averaged total queue length is about 18 packets, which is nearly halved compare with MaxWeight algorithm. If we increase β close to 1, for example 0.8, the average total queue length is nearly 8 packets, but the source rate cannot converge to optimal throughput under this circumstance. In fact, the convergence speed for source rate is even slower than MaxWeight algorithm (Figure 4.1 and Figure 4.2).

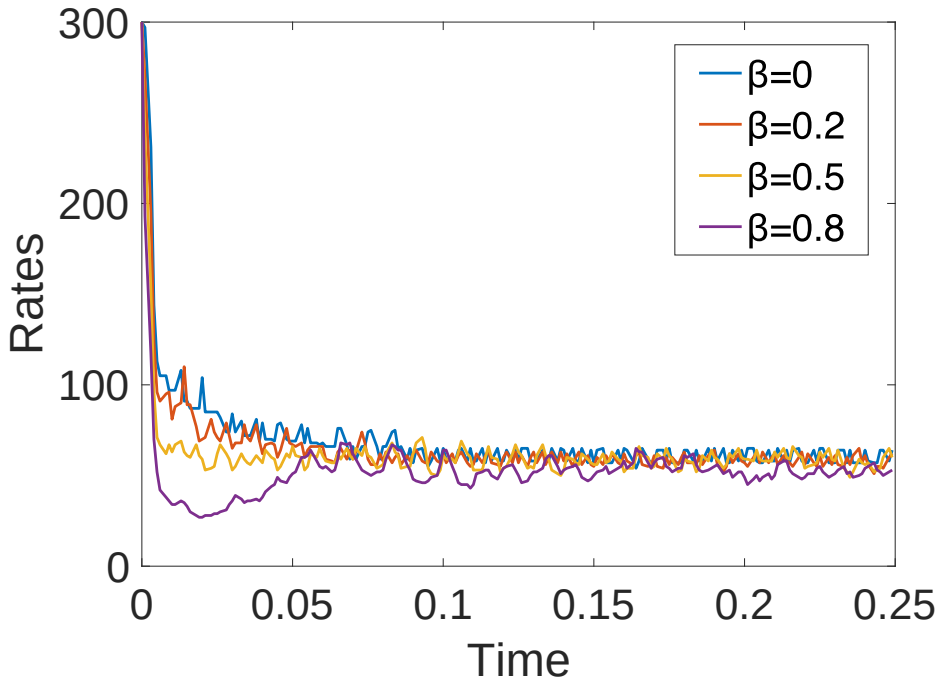

Figure 4.1 Source rate vs time

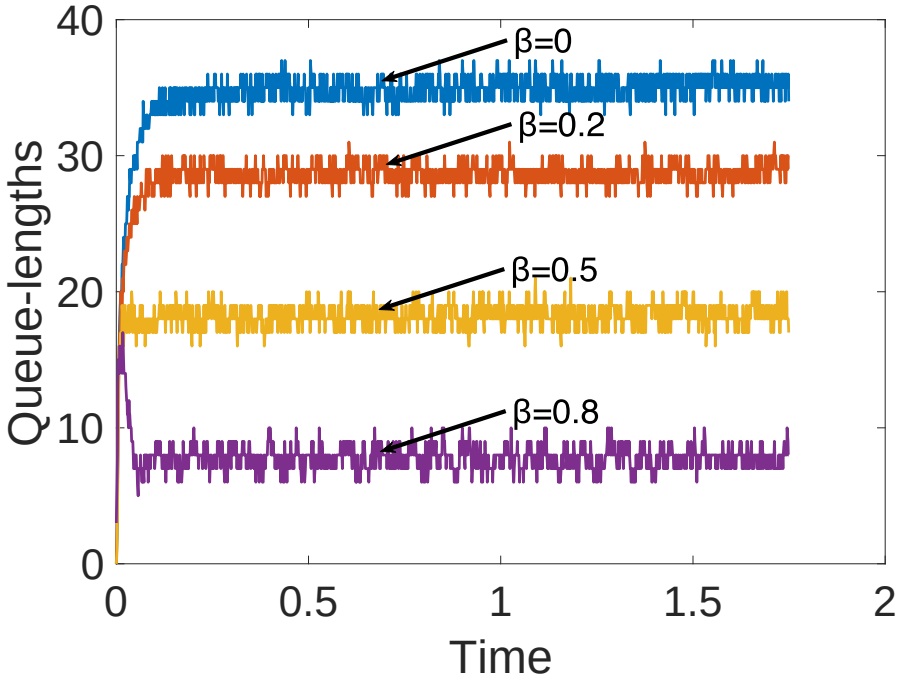

Figure 4.2 Queue length vs time

CHAPTER 5. CONCLUSION

5.1 Overview of Contributions

Cross-layer optimization is a new topic and contains a huge potential. But without a good emulating platform support, most great researches have stopped on the theoretical part. My work is the first in history to help wireless networking researchers test their cross-layer optimization algorithms in an emulation system.

By adding the Pathloss holder, MaxWeight scheduler and D-MGEN, researchers can use EMANE to emulate the cross-layer algorithm in Single-hop model. I also developed the centralized scheduler for applying cross-layer algorithm in Multi-hop networking. Contribute to my work, I successfully tested the MaxWeight and Heavyball algorithm in EMANE and obtain the performance data.

5.2 Future Work

The first thing I want to improve is to replace my centralized scheduler by a distributed scheduler. From it, I can save the channel resources and slot overhead during transmitting. Second, CORE is a very popular level 3 emulator which has an elegant GUI. It supports the use of EMANE on emulating level 1 and level 2 protocols. My next goal is to integrate my QLA implementation in EMANE to work with CORE GUI, which should provide a huge advance for researchers who want to study and test QLA algorithms.

REFERENCES

[1] "Extendable Mobile Ad-hoc Network Emulator (EMANE)," U.S. Naval Research Laboratory, [Online]. Available: https://www.nrl.navy.mil/itd/ncs/products/emane. [Accessed 21 8 2018].

[2] A. L. Russell, "OSI: The Internet That Wasn't," 30 07 2013. [Online]. Available: https://spectrum.ieee.org/tech-history/cyberspace/osi-the-internet-that-wasnt.

[3] J. L. Olenewa and M. D. Ciampa, Guide to Wireless Communications (2nd ed.), Boston, United States: THOMSON COURSE TECHNOLOGY, 2007.

[4] G. Miao, J. Zander, K. W. Sung and B. Slimane , Fundamentals of Mobile Data Networks, Cambridge, United Kingdom: Cambridge University Press, 2016.

[5] L. Tassiulas and A. Ephremides, "Jointly optimal routing and scheduling in packet ratio networks," *IEEE Transactions on Information Theory,* vol. 38, no. 1, pp. 165-168, 1992.

[6] X. Lin, N. B. Shroff and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE Journal on Selected areas in Communications,* vol. 24, no. 8, pp. 1452-1463, 2006.

[7] J. Liu, A. Eryilmaz, N. B. Shroff and E. S. Bentley, "Heavy-ball: A new approach to tame delay and convergence in wireless network optimization," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, San Francisco, 2016.

[8] V. Jacobson, "Congestion avoidance and control," *ACM SIGCOMM computer communication review,* vol. 18, no. 4, pp. 314-329, 1988.

[9] "Multi-Generator (MGEN)," U.S. Naval Research Laboratory, [Online]. Available: https://www.nrl.navy.mil/itd/ncs/products/mgen. [Accessed 21 8 2018].

[10] C. Joo, X. Lin and N. B. Shroff, "Greedy Maximal Matching: Performance Limits for Arbitrary Network Graphs Under the Node-Exclusive Interference Model," *IEEE Transactions on Automatic Control,* vol. 54, no. 12, pp. 2734 - 2744, 2009.