Fluid-based Analysis and Simulation of Internet Congestion Control Mechanisms

July 2010

Yusuke SAKUMOTO

Fluid-based Analysis and Simulation of Internet Congestion Control Mechanisms

Submitted to
Graduate School of Information Science and Technology
Osaka University

July 2010

Yusuke SAKUMOTO

List of Publications

Journal Papers

1. Satoshi Hasegawa, Yuske Sakumoto, Mirai Wakabayashi, Hiroyuki Ohsaki, and Makoto Imase, "Delay performance analysis on ad-hoc delay tolerant broadcast network," *IEICE Transactions on Communications Special Section on Ad Hoc and Mesh Networking for Next Generation Access Systems*, vol. E92-B, pp. 728–739, May 2009.

2. Yusuke Sakumoto, Hiroyuki Ohsaki, and Makoto Imase, "Stability analysis of XCP (eXplicit Control Protocol) with heterogeneous flows," *IEICE Transactions on Communications*, vol. E92-B, pp. 3174–3182, October 2009.

3. Yusuke Sakumoto, Hiroyuki Ohsaki, and Makoto Imase, "Improving robustness of XCP (eXplicit Control Protocol) for dynamic traffic," re-submited to *IEICE Transactions on Communications*, June 2010. (Conditionally Accepted)

4. Yusuke Sakumoto, Hiroyuki Ohsaki, and Makoto Imase, "Effectiveness of Thorup's Shortest Path Algorithm for Large-Scale Network Simulation," submited to *IEICE Transactions on Communications*, June 2010.

Refereed Conference Papers

1. Yusuke Sakumoto, Hiroyuki Ohsaki, and Makoto Imase, "On XCP stability in a heterogeneous network," in *Proceedings of 12th IEEE Symposium on Computers and Communications(ISCC'07)*, pp. 531–537, July 2007.

2. Yusuke Sakumoto, Ryouta Asai, Hiroyuki Ohsaki, and Makoto Imase, "Design and implementation of flow-level simulator for performance evaluation of large scale networks," in *Proceedings of 15th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2007)*, pp. 166–172, Octover 2007.

3. Yusuke Sakumoto, Hiroyuki Ohsaki, and Makoto Imase, "Increasing robustness of XCP (eXplicit Control Protocol) for dynamic traffic," in *Proceedings of 50th IEEE Global Telecommunications Conference (GLOBECOM 2007)*, pp. 2025–2030, November 2007.

4. Takeaki Nishioka, Yusuke Sakumoto, Hiroyuki Ohsaki, and Makoto Imase, "Design and implementation of flow-level simulator for a network with heterogeneous flows," in *Proceedings of 9th Annual International Symposium on Applications and the Internet (SAINT 2009)*, pp. 78–84, July 2009.

5. Shinpei Kuribayashi, Yuske Sakumoto, Satoshi Hasegawa, Hiroyuki Ohsaki, and Makoto Imase, "Performance evaluation of broadcast communication protocol DSCF (Directional Store-Carry-Forward) for VANETs with two-dimensional road model," in *Proceedings of IEEE*

International Workshop on Vehicular Communications, Networks, and Applications (VCNA 2009), pp. 615–619, December 2009.

6. Yusuke Sakumoto, Hiroyuki Ohsaki, and Makoto Imase, "On the effectiveness of thorup's shortest path algorithm for large-scale network simulation," to be presented at the *First Workshop on High Speed Network and Computing Environments for Scientific Applications (HSNCE 2010)*, July 2010.

Non-Refereed Technical Papers

1. Yusuke Sakumoto, Hiroyuki Ohsaki, and Makoto Imase, "Stability Analysis of XCP Connections with Different Propagation Delays," *Proceedings of IEICE General Conference 2006*, pp. 164, March 2006 (*in Japanese*).

2. Yusuke Sakumoto, Hiroyuki Ohsaki, and Makoto Imase, "Stability Analysis of Transport Protocol XCP for High-speed Networks," *Technical Report of IEICE* (IN2006-29), pp. 73–78, 2006 June (*in Japanese*).

3. Yusuke Sakumoto, Hiroyuki Ohsaki, and Makoto Imase, "Proposal of a Technique for Improving Robustness of Data Transfer Protocol XCP," *Technical Report of IEICE* (IN2006-91), pp. 13–18, November 2006 (*in Japanese*).

4. Ryouta Asai, Yusuke Sakumoto, Hiroyuki Ohsaki, and Makoto Imase, "Design and Implementation of Flow-Level Simulator for Large Scale Network Analysis," *Proceedings of IEICE General Conference 2007*, pp. 108, March 2007 (*in Japanese*).

5. Ryouta Asai, Yusuke Sakumoto, Hiroyuki Ohsaki, and Makoto Imase, Makoto IMASE, "Design and Implementation of Flow-Level Simulator for Performance Evaluation of Large Scale Networks," *Technical Report of IEICE* (IN2006-230), pp. 297–302, March 2007 (*in Japanese*).

6. Takeaki Nishioka, Yusuke Sakumoto, Hiroyuki Ohsaki, and Makoto Imase, "A Flow-Level Network Simulator Supporting Both Long-lived and Short-Lived TCP Flows," *Proceedings of IEICE General Conference 2008*, pp. 215, March 2008 (*in Japanese*).

7. Takeaki Nishioka, Yusuke Sakumoto, Hiroyuki Ohsaki, and Makoto Imase, "Design and Implementation of Flow-Level Simulator for a Network with Heterogeneous Flows," *Technical Report of IEICE* (IN2008-37), pp. 65–70, July 2008 (*in Japanese*).

8. Shinpei Kuribayashi, Yusuke Sakumoto, Satoshi Hasegawa, Hiroyuki Ohsaki, and Makoto Imase, "Characteristic Analysis of a Broadcast Communication Mechanism with Delay Tolerance for VANET," *Technical Report of IEICE* (IN2009-05), pp. 19–24, May 2009 (*in Japanese*).

Preface

In recent years, the scale of the Internet has been expanding rapidly. Because of widespread deployment and rapid advancement of Internet technologies, the number of hosts connected to the Internet and the capacity of the Internet have been increasing exponentially. Such explosive expansions of the Internet in both *size* and *speed* make it difficult to understand behavior of cogestion control mechanisms. Hence, performance evaluation technique of congestion control mechanisms for a large-scale network has been demanded by many networking researchers.

Application of mathematical analysis to performance evaluation of a large-scale network has been a hot topic among network researchers. In the past, a variety of research has been performed with regard to performance evaluation techniques for networks using mathematical analysis techniques. However, the networks in question are mostly small-scale.

On the other hand, several researchers try to enable simulation for a large-scale network, but there still remain several issues to be solved. Packet-level simulator has been widely used by many networking researchers. Packet-level simulator mimics behavior of every packet in a network. For instance, packet-level simulator simulates packet arrivals at a router and packet departures from a router. An advantage of packet-level simula-

tor includes its accuracy. Since packet-level simulator simulates behavior of every packet, packet-level performance metrics can be measured with packet-level simulator. On the contrary, a disadvantage of packet-level simulator is its inability to simulate large-scale networks. This is because the time complexity increases as the size and/or speed of a simulated network increases.

Fluid-based approaches of Internet congestion control mechanisms have been actively studied by many researchers. Essentially, fluid-based approaches approximate a series of packets as *a flow*, which enables application of mathematical analsysis and simulation such as differential equations and their numerical solutions. One of the noticeable features of fluid-based approaches is its analytical tractability as well as its powerfulness for accurate modeling of dynamical systems. The anothor noticeable features of fluid-based approaches mimics the flow-level behavior of the network. Flow-level bevaivior is lower granularity compared with packet-level behaivior.

In this thesis, to realize analysis and simulation of Internet congestion control mechanisms for large-scale networks, we tackle the issues with fluid-based approaches. Since fluid-based approaches have an excellent balance for detail and scalability, we expect that mathematical analysis and simulation of Internet congestion control mechanisms in large-scale network can be performed by utilizing fluid-based approaches. One of the noticeable features of fluid-based approaches is its analytical tractability as well as its powerfulness for accurate modeling of dynamical systems. Specifically, the following challenges are tackled.

First, we analyze the stability of XCP in a network with heterogeneous XCP flows (i.e., XCP flows with different propagation delays). Specifically,

we model a network with heterogeneous XCP flows using fluid-flow approximation. We then derive the conditions that XCP control parameters should satisfy for stable XCP operation.

Next, we propose an XCP-IR (XCP with Increased Robustness) that operates efficiently for dynamic XCP and non-XCP traffic. XCP-IR prevents instability of the XCP control caused by non-XCP traffic dynamics while preventing loss of the bottleneck link utilization caused by XCP traffic dynamics. We analyze stability and transient state performance of XCP-IR using the analytic approach.

Thirdly, we propose a flow-level simulator called FSIM (Fluid-based SIMulator) for performance evaluation of large-scale networks, and verify its effectiveness using our FSIM implementation. The notable feature of our flow-level simulator FSIM is fast simulation execution compared with a conventional flow-level simulator. For accelerating simulation execution, our flow-level simulator FSIM adopts an adaptive numerical computation algorithm for ordinary differential equations. Another features of our flow-level simulator FSIM are accuracy and compatibility with an existing network performance analysis tool. For improving simulation accuracy, our flow-level simulator FSIM utilizes accurate fluid-flow models.

Finally, we investigate the effectiveness of Thorup's algorithm for large-scale network simulation. One of challenges toward realization of large-scale network simulation is efficient execution of shortest-path routing. The time complexity for solving a single-source shortest path (SSSP) problem using Dijkstra's algorithm with a binary heap is $O((E + N) \log N)$. An efficient linear time algorithm called *Thorup's algorithm* with the time complexity of $O(E + N)$ has been proposed. Through extensive experiments with our implementations of Thorup's and Dijkstra's algorithms, we com-

pare the performance (i.e., the execution time and memory consumption) of those algorithms.

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor, Professor Makoto Imase of Osaka University. He has been carefully advising me for seven years since I had joined his laboratory. He always leads me in the right direction by his sophisticated thinking. His advice and comments have enabled me to grow up.

I would like to express my gratitude to Professor Hirotaka Nakano, Professor Teruo Higashino, Professor Koso Murakami, Professor Masayuki Murata of Osaka University for their valuable comments and advice, and reviewing my thesis. Especially, I would like to express my deepest appreciation to Professor Masayuki Murata for his advice and suggestion when I started research activity.

I would like to give heartfelt thanks to Associate Professor Hiroyuki Ohsaki. He has been actual advisor for my research works. I could not conduct my research works without his support and leadership. When I was worried about something, he always gave me a helping hand with tender toughness. What I have learned from him would become very valuable in my life.

I am thankful to Dr. Hideyuki Shimonishi, Principle Researcher, NEC Co., Ltd, for his valuable comments. His comments raise the quality of my

research.

I would like to very grateful to Dr. Satoshi Hasegawa, President, Techno-Edge Co., Ltd., for teaching me the life for a member of society.

I wish to express to thank to Associate Professor Harumasa Tada of Kyoto University of Education and Assistant Professor Osamu Honda of Onomichi University for their support when I was a master course student.

Special thanks to all laboratory members for support. Specifically, I would like to thank to Assistant Professor Yuki Koizumi for his right advice. I thank to Sho Tsugawa, Takamichi Nishijima, Keiichiro Tsukamoto, Keita Shigemori and Shinpei Kuribayashi for their warm encouragement.

Finally, I would like to greatly appreciated to my wife, Kana, my parents, Kazue and Taeko, my grandmother, Tomoko, and my sister, Chie for their heartily help and support on my work.

Without all of these, I could not complete this work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

In recent years, the scale of the Internet has been expanding rapidly. Because of widespread deployment and rapid advancement of Internet technologies, the number of hosts connected to the Internet and the capacity of the Internet have been increasing exponentially [1-3]. Such explosive expansions of the Internet in both *size* and *speed* make it difficult to understand behavior of cogestion control mechanisms [4-6]. Hence, performance evaluation technique of congestion control mechanisms for a large-scale network has been demanded by many networking researchers [7, 8].

Performance evaluation techniques for communication networks are classified into three categories: mathematical analysis, simulation, and experiment [9].

Mathematical analysis is a technique for performance evaluation utilizing a mathematical model of the network under study. Mathematical analysis is generally suitable for analyzing characteristics of a congestion control mechanism as a feedback-based control system, such as stability

and transient performance.

Simulation is a common technique for performance evaluation utilizing computers. In simulation, computer models of building blocks of the network under study are built, and behavior of those building blocks are simulated [9]. Compared with mathematical analysis, simulation can be applied to performance evaluation of rather complicated networks.

Experiment is a technique for performance evaluation utilizing a real system [9]. In experiment, the network under study is constructed using real devices and computers. Although experiment enables detailed performance evaluation, it generally lacks flexibility and also requires significant amount of cost for building the real system. For performance evaluation of a large-scale network, experiment requires a number of network devices and computers, making it unrealistic to apply to a large-scale network.

Considering cost required for performance evaluation, mathematical analysis and simulation are reasonable approaches for performance evaluation of congestion control mechanisms in large-scale networks.

Application of mathematical analysis to performance evaluation of a large-scale network has been a hot topic among network researchers [10, 11]. In the past, a variety of research has been performed with regard to performance evaluation techniques for networks using mathematical analysis techniques. However, the networks in question are mostly small-scale.

On the other hand, several researchers try to enable simulation for a large-scale network [12], but there still remain several issues to be solved. Packet-level simulator has been widely used by many networking researchers. Packet-level simulator mimics behavior of every packet in a network [13]. For instance, packet-level simulator simulates packet arrivals at a router and packet departures from a router. An advantage of packet-level simula-

tor includes its accuracy [14]. Since packet-level simulator simulates behavior of every packet, packet-level performance metrics can be measured with packet-level simulator. On the contrary, a disadvantage of packet-level simulator is its inability to simulate large-scale networks. This is because the time complexity increases as the size and/or speed of a simulated network increases [14].

Fluid-based approaches of Internet congestion control mechanisms have been actively studied by many researchers [15-18]. Essentially, fluid-based approaches approximate a series of packets as *a flow*, which enables application of mathematical analsysis and simulation such as differential equations and their numerical solutions. One of the noticeable features of fluid-based approaches is its analytical tractability as well as its powerfulness for accurate modeling of dynamical systems. The anothor noticeable features of fluid-based approaches mimics the flow-level behavior of the network. Flow-level behaivior is lower granularity compared with packet-level behaivior.

Using fluid-based approaches, application of mathematical analysis and simulation to performance evaluation of congestion control mechanisms in a large-scale network become possible. However, fluid-based approaches are still immature so that several issues must be solved for applying to fluid-based analyis of complex network protocols such as Internet congestion control mechanisms for high-speed networks, and develpping the efficient numerical computation algorithm.

1.2 Internet Congestion Control Mechanisms

TCP (Transmission Control Protocol) has being widely used in the Internet to carry data traffic [19]. There are various versions of TCP, and the most popular ones are TCP version Reno (TCP Reno) and its variants [20]. Several problems of TCP Reno have been reported such as its inability to support rapidly increasing speeds of recent networks [21-23].

One of the serious problems with TCP Reno is that a large number of packets sent into the network are discarded. This problem is caused by inability of source hosts to detect congestion before some packets are lost, and large delay for source hosts to detect congestion when the round-trip time between end hosts is large. Since the scale and the speed of network is continuously increasing, it is expected that the performance of TCP Reno would be further degraded due to an increased number of packet losses before source hosts' congestion detection.

For solving problems of TCP Reno in a high-speed and wide-area network, many congestion control mechanisms (router-assistent congestion control mechanism) using explicit feedback from a router to end hosts are proposed [24-27]. Compared with TCP Reno, these protocols perform efficient congestion control between end hosts with the aid of routers. As a router-assisted congestion control mechanism, XCP (eXplicit Control Protocol) has been proposed [24, 28]. XCP is a sort of window-based flow control mechanisms. An XCP router periodically calculates the amount of window size increase/decrease for a source host, and notifies source hosts of it as explicit feedback. With such explicit feedback, an XCP source host can quickly and appropriately respond to congestion status of the network.

A router-assisted congestion control mechanism has advantage and dis-

advantage. An end-to-end congestion control mechanism estimates a congestion status of a network only using the information that can obtain at the end host. On the other hand, a router-assisted congestion control mechanism estimates a congestion status of a network utilizing feedback information from routers. An advantage of router-assisted congestion control mechanisms is ability to accurately control the congestion compared with end-to-end congestion control mechanisms. Hence, it is expected that a router-assisted congestion control mechanism realizes high performance compared with end-to-end congestion control mechanisms. A disadvantage of router-assisted congestion control mechanisms is needed to replace routers in the network. Researches for incremental deployment of XCP have been performed [24, 28, 29]. By utilizing these research results, it is possible to deploy XCP. To realize high performance in a large-scale network, it would be better to actively utilize the feedback information from a router like router-assisted congestion control mechanisms. Therefore, we focus on the router-assisted congestion control mechanism XCP.

1.3 Fluid-based Analysis and Simulation

There exist a few analytical studies on XCP, all of which used fluid-flow approximation [24, 15, 30, 16]. In [24, 15], by assuming an identical propagation delay for all XCP flows, stability of XCP has been analyzed. The authors of [24] have derived a sufficient condition for XCP control parameters to stabilize XCP's operation. By extending the analytic model in [24], the authors of [15] have shown that operation of XCP becomes unstable when the available bandwidth of an XCP router's output link is varied. The authors of [30] have analyzed stability of XCP using a Lyapunov func-

tion. In [30], the authors mention possibility of extending their stability analysis to handle heterogeneous XCP flows, but the details are not presented. In [16], steady state performance of XCP in a tandem network (i.e., a network with multiple routers and heterogeneous flows) has been analyzed. Specifically, the authors of [16] have derived throughput of XCP flows in steady state, and have shown that fairness among XCP flows is significantly degraded unless control parameters of an XCP router are configured appropriately. Although multiple XCP flows are modeled in [16] for analyzing fairness among XCP flows, stability of XCP has not been investigated.

Flow-level simulator mimics behavior of every flow in a network [31]. For instance, packet arrivals at a router and packet departures from a router are aggregated as a flow (i.e., a series of packets) in flow-level simulator. Since flow-level behavior is lower granularity compared with packet-level behavior, flow-level simulator has the following advantage and disadvantage. An advantage of flow-level simulator is small time complexity compared with packet-level simulator. Hence, it is expected that flow-level simulator can simulate a large-scale network, where the number of in-flight packets is enormous. On the contrary, disadvantages of flow-level simulator are (1) low accuracy compared with packet-level simulator, and (2) inability to measure packet-level performance metrics. Therefore, the application region of flow-level simulator is restricted to the network that a highly accurate fluid-flow approximation model can be build, and it is sufficient to measure flow-level performance metrics for evaluate performance of.

Flow-level simulation can be utilized to measure the flow level performance metrics of a large-scale network to be hardly simulated by packet-

level simulator. In particular, if the target network to simulate by flow-level simulator is a wired network, it may expect to realize highly accurate simulation by using the fluid-flow approximation model studied so far. Moreover, since many of packets transfered in a wired network are data packets, in the performance evaluation of a wired network, measuring flow-level performance metrics (e.g., throughput and round-trip time), which it is important for data transfer, is often sufficient. Therefore, flow level simulator would be sufficient to evaluate the performance of a large-scale wired network.

1.4 Outline of Thesis

In this thesis, to realize analysis and simulation of Internet congestion control mechanisms for large-scale networks, we tackle the issues with fluid-based approaches. Since fluid-based approaches have an excellent balance for detail and scalability, we expect that mathematical analysis and simulation of Internet congestion control mechanisms in large-scale network can be performed by utilizing fluid-based approaches. One of the noticeable features of fluid-based approaches is its analytical tractability as well as its powerfulness for accurate modeling of dynamical systems. Specifically, the following challenges are tackled.

Stability Analysis of XCP (eXplicit Control Protocol) with Heterogeneous Flows [32-34]

In Chapter 2, we analyze the stability of XCP in a network with heterogeneous XCP flows (i.e., XCP flows with different propagation delays). Specifically, we model a network with heterogeneous XCP flows using fluid-

flow approximation. We then derive the conditions that XCP control parameters should satisfy for stable XCP operation.

Through several numerical examples and simulation results, we quantitatively investigate effect of system parameters and XCP control parameters on stability of the XCP protocol. Our findings include: (1) when XCP flows are heterogeneous, XCP operates more stably than the case when XCP flows are homogeneous, (2) conversely, when variation in propagation delays of XCP flows are large, operation of XCP becomes unstable, and (3) the output link bandwidth of an XCP router is independent of stability of the XCP protocol.

The primary contribution in this chapter is to mathematically reveal stability properties of the XCP protocol with heterogeneous flows.

Improving Robustness of XCP (eXplicit Control Protocol) for Dynamic Traffic [35-37]

In Chapter 3, we propose an XCP-IR (XCP with Increased Robustness) that operates efficiently for dynamic XCP and non-XCP traffic. XCP-IR prevents instability of the XCP control caused by non-XCP traffic dynamics while preventing loss of the bottleneck link utilization caused by XCP traffic dynamics.

In Chapter 3, we analyze stability and transient state performance of XCP-IR using the analytic approach in Chapter 2. We evaluate the effectiveness of XCP-IR through extensive simulations. Through investigation of both steady state and transient state performances, we show that XCP-IR operates efficiently even for dynamic traffic.

Design and Implementation of Flow-Level Simulator FSIM for Performance Evaluation of Large Scale Networks [38]

In Chapter 4, we propose a flow-level simulator called FSIM (Fluid-based SIMulator) for performance evaluation of large-scale networks, and verify its effectiveness using our FSIM implementation. The notable feature of our flow-level simulator FSIM is fast simulation execution compared with a conventional flow-level simulator [39]. For accelerating simulation execution, our flow-level simulator FSIM adopts an adaptive numerical computation algorithm for ordinary differential equations. Another features of our flow-level simulator FSIM are accuracy and compatibility with an existing network performance analysis tool. For improving simulation accuracy, our flow-level simulator FSIM utilizes accurate fluid-flow models [40]. Also, the flow-level simulator FSIM can input and output files compatible with ns-2 [13], which is one of the most popular packet-level simulators.

In Chapter 4, through extensive experiments using our FSIM implementation, we evaluate the effectiveness of our flow-level simulator FSIM in terms of simulation speed, accuracy and memory consumption. Consequently, we show that our flow-level simulator FSIM outperforms a conventional flow-level simulator; i.e., it realizes approximately 200%-300% faster simulation with higher accuracy and less memory consumption than a conventional flow-level simulator.

Effectiveness of Thorup's Shortest Path Algorithm for Large-Scale Network Simulation [41, 42]

Since one of the challenges toward realization of large-scale network simulation is efficient execution of shortest-path routing (e.g., static routing) [43-

45], in Chapter 5, we investigate the effectiveness of Thorup's algorithm for a large-scale network simulation.

All network simulators require some type of routing to be performed during its simulation. Network simulators generally use a solution for a single-source shortest path (SSSP) problem for static routing. For instance, one of the most popular network simulators, ns-2 [13], uses Dijkstra's algorithm with a binary heap (hereafter referred to as DIJKSTRA-BH) [46, 47]. With static routing, ns-2 first obtains shortest paths for all source–destination node pairs.

The time complexity of DIJKSTRA-BH is $O((E+N)\ \log N)$ where N and E are the number of vertices and edges in a graph [46, 47]. Since network simulators usually need to obtain shortest paths for all source–destination node pairs, network simulators consume significant amount of time just for static routing [43].

An efficient solution for a single-source shortest path problem called *Thorup's algorithm* has been proposed [48]. The original Thorup's algorithm (hereafter THORUP-FR) uses Fredman's algorithm [49] for obtaining the minimum spanning tree. THORUP-FR is a linear time algorithm with the time complexity of $O(E + N)$. A simplified version of Thorup's algorithm (hereafter THORUP-KL), which uses Kruskal's algorithm [50] for obtaining the minimum spanning tree, is also discussed in [48]. The time complexity of THORUP-KL is $O(E\,\alpha(N) + N)$ where $\alpha(N)$ is the functional inverse of the Ackerman function [47].

Comparison of time complexities of Thorup's and Dijkstra's algorithms suggests that Thorup's algorithm seems to be more efficient than Dijkstra's for, in particular, large-scale network simulation. However, to the best of our knowledge, effectiveness of Thorup's algorithm compared with Dijk-

stra's algorithm for large-scale network simulation has not been fully investigated.

The objective of this chapter is therefore to investigate the effectiveness of Thorup's algorithm by comparing with Dijkstra's, and to answer the following questions.

1. How efficiently/inefficiently does Thorup's algorithm perform compared with Dijkstra's for large-scale (e.g., million nodes) network simulation?

2. How and why does the practical performance of Thorup's and Dijkstra's algorithms deviate from their time complexities (i.e., theoretical performance)?

In the pioneering work by Asano *et al.* [51], it has been reported that practical efficiency of THORUP-FR is considerably lower than that of Dijkstra's algorithm with a Fibonatti heap (hereafter DIJKSTRA-FH) [52]. DIJKSTRA-FH has a better amortized running time than DIJKSTRA-BH [46]. In [51], the authors have investigated the performance of THORUP-FR (i.e., the *original* Thorup's algorithm) using their implementation for medium-scale random graphs with 50,000 vertices. Even with the linear time complexity of THORUP-FR, the authors have shown that THORUP-FR is at least 10 times slower than DIJKSTRA-FH. The authors explain that their implementation of THORUP-FR is very slow due to the time for constructing the minimum spanning tree with Fredman's algorithm [51].

In this chapter, we intentionally use THORUP-KL (i.e., a *simplified* version of Thorup's algorithm) with the time complexity of $O(E\,\alpha(N) + N)$.

Through extensive experiments with our implementations of THORUP-KL and DIJKSTRA-BH, we compare their performances (i.e., the execution

time and memory consumption). Our findings include that (1) THORUP-KL is almost always faster than DIJKSTRA-BH for large-scale network simulation, and (2) the performances of THORUP-KL and DIJKSTRA-BH deviate from their time complexities due to the existence of memory cache in the microprocessor.

Chapter 2

Stability Analysis of XCP (eXplicit Control Protocol) with Heterogeneous Flows

2.1 XCP (eXplicit Control Protocol)

In this section, the congestion control algorithm of XCP is briefly summarized. Refer to [24] for details of the XCP algorithm.

In XCP, congestion information is exchanged between a source host and a router using *congestion header* in a packet. An overview of the XCP congestion control using the congestion header of a packet is illustrated in Fig. 2.1.

XCP is a sort of window-based flow control mechanisms. In XCP, a router calculates the amount of window size increase/decrease for a source host, and it then notifies the source host of the calculated value as explicit feedback. The congestion header of a packet stores information on a source host and a router: e.g., the window size and the estimated round-trip time

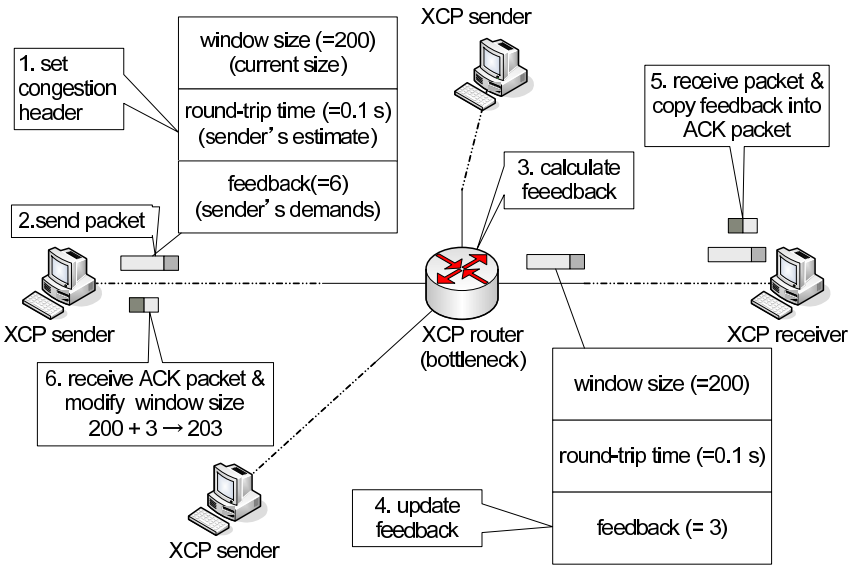


Figure 2.1: Overview of XCP congestion control algorithm; a router calculates the amount of window size increase/decrease for a source host, and it then notifies the source host of the calculated value as explicit feedback.

of the source host, and the amount of window-size increase/decrease (i.e., feedback value) calculated by the router.

At the time of packet transmission, a source host stores its estimated round-trip time, its current window size, and the initial value of the feedback value (i.e., the amount of window size increase requested by the source host) in the congestion header of the packet. This enables the XCP router to know the status of the source host.

When the packet arrives at an XCP router, the router calculates a feedback value based on the information stored in the congestion header of the packet. The router overwrites the feedback value in the congestion header of the packet with the calculated feedback value, if the feedback value stored in the congestion header is larger than the calculated feedback value. The XCP router then forwards the packet to its downstream router.

Once the packet arrives at a destination host, the destination host returns an ACK (ACKnowledgement) packet to the source host. At this time, the congestion header of the data packet is copied to the congestion header of the ACK packet. This makes it possible for the source host to know the congestion information of XCP routers by way of the destination host.

Finally, when the source host receives the ACK packet, the feedback value stored in the congestion header of the ACK packet is added to the current window size of the source host.

In what follows, we explain how an XCP router calculates a feedback value (i.e., the amount of increase/decrease of the window size of a source host).

The control mechanism of an XCP router is composed of *efficiency controller*, which tries to maximize utilization of the router, and *fairness controller*, which tries to realize fairness among competing XCP flows. The efficiency controller and the fairness controller are invoked every average round-trip time of all XCP flows. The efficiency controller calculates the total amount of rate increase/decrease for all XCP flows. The fairness controller then calculates the amount of rate increase/decrease for each XCP flow. An XCP router calculates a feedback value based on the amount of rate increase/decrease calculated by the fairness controller and information stored in the congestion header of arriving packets. In what follows, algorithms of the efficiency controller and the fairness controller are briefly explained.

The efficiency controller calculates the aggregate feedback value ϕ (i.e., the total amount of rate increase/decrease for all XCP flows) from the packet

arrival rate at the XCP router and the current queue length as

$$\phi = \alpha\, d\, S - \beta\, Q, \tag{2.1}$$

where d is the average round-trip time of XCP flows accommodated in the XCP router, S is the available bandwidth of the link (i.e., the output link bandwidth excluding the current packet arrival rate), Q is the minimum queue length observed during the average round-trip propagation time, and α and β are control parameters of the XCP router.

The fairness controller distributes the aggregate feedback value ϕ to all XCP flows. The fairness controller realizes fairness among XCP flows by performing an AIMD (Additive Increase and Multiplicative Decrease) control. Namely, if $\phi \geq 0$, the fairness controller allocates ϕ so that the increase in throughput of all XCP flows is the same. On the contrary, if $\phi < 0$, the fairness controller allocates ϕ so that the decrease in throughput of a XCP flow is proportional to its current throughput. Specifically, the fairness controller calculates ξ_p and ξ_n, which are used for calculating the feedback value.

$$\xi_p = \frac{h + [\phi]^+}{d \sum_{m=1}^{N} \frac{rtt_m\, s_m}{w_m}} \tag{2.2}$$

$$\xi_n = \frac{h + [-\phi]^+}{d\, T} \tag{2.3}$$

In the above equations, N is the number of packets arrived at the XCP router during the average round-trip time, and T is the total size of packets arrived during the average round-trip time.

Also, w_m and rtt_m are the window size and the estimated round-trip time stored in the congestion header of the m-th packet of N packets ar-

rived during the average round-trip time, and s_m is the packet size of the m-th packet. Note that $[x]^+ \equiv \max(x, 0)$.

In Eq. (2.3), h is called *shuffle traffic*, and is determined by

$$h = [\gamma T - |\phi|]^+, \tag{2.4}$$

where γ is a control parameter of an XCP router.

Finally, an XCP router calculates the feedback value $H_{feedback}$ for the m-th packet as

$$H_{feedback} = \xi_p \frac{rtt_m^2 s_m}{w_m} - \xi_n rtt_m s_m. \tag{2.5}$$

2.2 Modeling with Fluid-Flow Approximation

In this chapter, we model a network with heterogeneous XCP flows with different propagation delays (i.e., XCP flows traversing links with different propagation delays) sharing the single bottleneck link as a discrete-time system (Fig. 2.2). XCP flows are classified into *flow classes*, in which XCP flows have the identical propagation delay. In our analysis, dynamics of transfer rates from XCP flows and the queue length of an XCP router are modeled as discrete-time models with slot length of Δ. The definition of symbols used throughout our analysis is summarized in Tab. 2.1.

In this chapter, we focus on the stability of XCP in a network with a single XCP router, rather than in a tandem network. We believe that it is possible to extend our analytic approach to a tandem network with multiple XCP routers, but it is beyond the scope of this chapter. It should be noted that the operation of XCP may become unstable in a tandem network

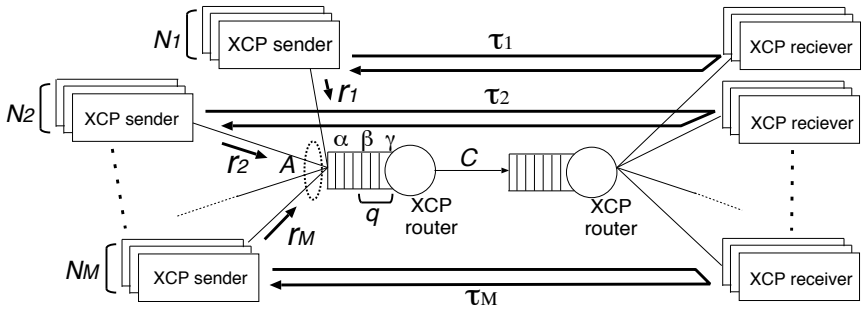


Figure 2.2: Analytic model; heterogeneous XCP flows with different propagation delays share the single bottleneck.

when many XCP flows are frequently activated and/or deactivated [53]. The main purpose of this chapter is, however, to investigate the effect of heterogeneous flow on the stability of the XCP protocol.

In what follows, using a fluid-based modeling approach, we derive a detailed model of a network with heterogeneous XCP flows.

First, we model dynamics of the transfer rate of an XCP flow. In our analysis, we assume: (a) all XCP flows with the same propagation delay synchronize, (b) all XCP source hosts always have data to transfer, (c) sizes of all packets are equal, (d) the window size of a source host is changed only by receiving the feedback value from XCP routers (i.e., effect of timeouts triggered by a large number of packet losses are negligible), and (e) the round-trip time of an XCP flow is equal to its two-way propagation delay.

Since an XCP router performs the same congestion control for all XCP flows with the same round-trip time (see Eq. (2.5)), the assumption (a) is reasonable. Moreover, since XCP is mainly used for transferring a large amount of data in a high-speed network, assumptions (b) through (d) should be appropriate. The assumption (e) is reasonable since the control objective of an XCP router is to minimize its queue length, resulting negligible queu-

Table 2.1: The definition of symbols

M	the number of flow classes
N_i	the number of XCP flows in flow class i
Δ	slot length
r_i	transfer rate of XCP flows in flow class i
τ_i	two-way propagation delay of XCP flows in flow class i
d	average round-trip time of all XCP flows
s	packet size
T	the total size of packets arrived during the average round-trip time
q	current queue length of XCP router
ϕ	aggregate feedback of XCP router
h	shuffle traffic of XCP router
A	packet arrival rate at XCP router
C	output link bandwidth of XCP router
α	XCP control parameter
β	XCP control parameter
γ	XCP control parameter

ing delay at the router buffer. Note that in our analysis, all XCP flows are assumed to start their transmissions simultaneously from an initial state. Hence, the queue length of the XCP router is not likely to be overloaded, which validates the assumption (e).

The transfer rate and the two-way propagation delay of XCP flows in flow class i are denoted by r_i and τ_i, respectively. Moreover, the number of XCP flows in flow class i is denoted by N_i. Then, the packet arrival rate A at an XCP router and the average round-trip time d of all XCP flows are given by

$$A = \sum_{i=1}^{M} N_i \, r_i, \tag{2.6}$$

$$d = \frac{\sum_{i=1}^{M} N_i \, \tau_i}{\sum_{i=1}^{M} N_i}. \tag{2.7}$$

Since $\sum_{m=1}^{N} rtt_m \, s_m / w_m \simeq d \sum_{i=1}^{M} N_i$ in Eq. (2.2) [16] and $T \simeq A \, d$, ξ_p

and ξ_n (Eqs. (2.2) and (2.3)) are given by

$$\xi_p = \frac{h + [\phi]^+}{d^2 \sum_{i=1}^{M} N_i}, \tag{2.8}$$

$$\xi_n = \frac{h + [-\phi]^+}{d^2 A}. \tag{2.9}$$

In the above equations, the aggregate feedback value ϕ and the shuffle traffic h are given by Eqs. (2.1) and (2.4) as

$$h = [\gamma\, d\, A - |\phi|]^+, \tag{2.10}$$

$$\phi = \alpha\, d\, (C - A) - \beta\, q, \tag{2.11}$$

where q is the current queue length of the XCP router, and C is the output link bandwidth of the XCP router.

From Eqs. (2.8) and (2.9), the feedback value $H_{feedback_i}$ for XCP flows in flow class i is given by

$$H_{feedback_i} = \frac{h + [\phi]^+}{d^2 \sum_{j=1}^{M} N_j} \frac{\tau_i\, s}{r_i} - \frac{h + [-\phi]^+}{d^2 A}\, \tau_i\, s. \tag{2.12}$$

Hence, at the time of ACK packet reception, the amount of change in the transfer rate of XCP flows in flow class i is given by

$$\frac{H_{feedback_i}}{\tau_i} = \frac{h + [\phi]^+}{d^2 \sum_{j=1}^{M} N_j} \frac{s}{r_i} - \frac{h + [-\phi]^+}{d^2 A} s. \tag{2.13}$$

The transfer rate of XCP flows in flow class i and the current queue length of the XCP router at slot k are denoted by $r_i(k)$ and $q(k)$, respectively. Without loss of generality, we assume that the propagation delay from a source host to the XCP router is zero, and that the propagation delay from the XCP router to source hosts by way of the destination hosts is τ_i. The

information stored in the congestion header of the ACK packet received by a source host at slot k is $k - \tau_i/\Delta$ slots old. Moreover, the number of ACK packets that a source host receives during the slot length Δ can be approximated by $r_i(k - \tau_i/\Delta)\,\Delta/s$. Thus, from Eq. (2.13), the transfer rate of XCP flows in flow class i at $(k+1)$-th slot is given by

$$r_i(k+1) \simeq r_i(k) + \Delta \frac{h(k - \frac{\tau_i}{\Delta}) + [\phi(k - \frac{\tau_i}{\Delta})]^+}{d^2 \sum_{j=1}^{M} N_j} - \Delta \frac{r_i(k - \frac{\tau_i}{\Delta}) \left(h(k - \frac{\tau_i}{\Delta}) + [-\phi(k - \frac{\tau_i}{\Delta})]^+\right)}{d^2\, A(k - \frac{\tau_i}{\Delta})}. \tag{2.14}$$

Next, we model the dynamics of the queue length of an XCP router. Letting $q(k)$ be the current queue length of the XCP router at slot k, the current queue length $q(k+1)$ at slot $k+1$ is approximately given by

$$q(k+1) \simeq \begin{cases} q(k) + \Delta\ (A(k) - C) & \text{if } q(k) > 0 \\ q(k) + \Delta\ [A(k) - C]^+ & \text{if } q(k) = 0 \end{cases}. \tag{2.15}$$

2.3 Stability Analysis

In what follows, using the fluid-flow model of XCP derived in Section 2.2, we analyze the stability (local asymptotic stability) of XCP around its equilibrium point using the same analytic approach with [54]. In what follows, equilibrium values of the transfer rate $r_i(k)$ and the current queue length $q(k)$ are denoted by r_i^* and q^*, respectively. First, we linearize the fluid-flow model defined by Eqs. (2.14) and (2.15) at its equilibrium point. The aggregate feedback value $\phi(k)$ and the current queue length $q(k)$ are discontinuous at the equilibrium point (i.e., $\phi^* = 0$ and $q^* = 0$). For alleviating such a discontinuity problem, we introduce the following approximation for a

sufficiently small Δ.

$$\frac{[f(x+\Delta)]^+ - [f(x)]^+}{\Delta} \simeq \frac{1}{2}\frac{f(x+\Delta)-f(x)}{\Delta} \tag{2.16}$$

Thereby, Eqs. (2.14) and (2.15) can be approximated as

$$\begin{aligned} r_i(k+1) \simeq r_i(k) + \Delta\,\frac{h(k-\frac{\tau_i}{\Delta}) + \phi(k-\frac{\tau_i}{\Delta})/2}{d^2\sum_{j=1}^{M} N_j} \\ -\Delta\frac{r_i(k-\frac{\tau_i}{\Delta})\left(h(k-\frac{\tau_i}{\Delta}) - \phi(k-\frac{\tau_i}{\Delta})/2\right)}{d^2\,A(k-\frac{\tau_i}{\Delta})}, \end{aligned} \tag{2.17}$$

and

$$q(k+1) \simeq q(k) + \frac{\Delta\,(A(k)-C)}{2}. \tag{2.18}$$

Equations (2.17) and (2.18) suggest that state variables at slot $k+1$ are determined by state variables from $k-\nu$ ($\nu \equiv \max_{1\le i\le M}\tau_i/\Delta$) to slot k.

Furthermore, we linearize Eq. (2.17) around its equilibrium point as

$$\begin{aligned} r_i(k+1) \;\simeq\; & \sum_{m=1}^{M}\sum_{n=0}^{\nu}\frac{\partial r_i(k+1)}{\partial r_m(k-n)}\left\{r_m(k-n) - r_m^*\right\} \\ & + \sum_{n=0}^{\nu}\frac{\partial r_i(k+1)}{\partial q(k-n)}\left\{q(k-n) - q^*\right\}. \end{aligned} \tag{2.19}$$

We introduce a state vector $\mathbf{x}(k)$ that is composed of differences be-

tween each state variable at slot $k, \ldots, k-\nu$ and their equilibrium values.

$$\mathbf{x}(k) = \begin{pmatrix} r_1(k) - r_1^* \\ \vdots \\ r_1(k-\nu) - r_1^* \\ \vdots \\ r_M(k) - r_M^* \\ \vdots \\ r_M(k-\nu) - r_M^* \\ q(k) - q^* \\ \vdots \\ q(k-\nu) - q^* \end{pmatrix} \tag{2.20}$$

The relation between $\mathbf{x}(k)$ and $\mathbf{x}(k+1)$ can be represented using a state transition matrix $\mathbf{B}$ as

$$\mathbf{x}(k+1) = \mathbf{B}\,\mathbf{x}(k). \tag{2.21}$$

Note that the state transition matrix $\mathbf{B}$ is independent of the slot k. If $\mathbf{x}(k)$ converges to the zero vector for $k \to \infty$, the system is stable. Otherwise, the system is unstable. The state transition matrix $\mathbf{B}$ determines the stability of the system. Let $\lambda_i (1 \leq i \leq (M+1)(\nu+1))$ be the eigenvalues of the state transition matrix $\mathbf{B}$. The maximum absolute value of eigenvalues (i.e., maximum modules) determines the stability around its equilibrium point. It is known that the system is stable if the maximum modulus is less than unity [55]. Namely, if the maximum modulus is less than unity, $x(k)$ converges to the zero vector for $k \to \infty$.

Table 2.2: The parameter configuration use in numerical examples and simulation results

C	400 [Mbit/s]
τ_1	10 [ms]
τ_2	20 [ms]
N_1	10
N_2	10
γ	0.1

2.4 Numerical Examples and Simulation Results

2.4.1 Experimental design

In this section, through several numerical examples and simulation results, we investigate the effect of system parameters and XCP control parameters on stability of XCP protocol. Due to space limitation, in what follows, only results in the case of two flow classes ($M = 2$) are shown. Unless explicitly stated, the parameter configuration shown in Tab. 2.2 is used. The slot length is set to $\Delta = \min(\tau_1, \tau_2)$.

After examining various numerical examples of our stability analysis in Section 2.3, we found that the control parameter γ is hardly affected stability of the XCP protocol. In this chapter, we therefore focus only on the effect of control parameters α and β. [1]

Also, we found that the output link bandwidth C of an XCP router did not affect stability of the XCP protocol. Although the proof is not shown in this chapter due to space limitation, independence of the output link bandwidth C from stability of the XCP protocol can be confirmed from the fact that expansion of Eq. (2.19) eliminates all C's. This phenomenal finding will be confirmed through simulation experiments in Section 2.4.2.

[1] Note that it is shown in [16] that the control parameter γ does affect efficiency of the XCP router and fairness among XCP flows.

2.4.2 Effect of propagation delays

First, the effect of propagation delays of XCP flows on stability of the XCP protocol is investigated. Figure 2.3 shows stability regions of XCP control parameters (α β) for different settings of two-way propagation delays : i.e., (τ_1, τ_2) = (10, 10), (20, 20), (10, 20), and (10, 30) [ms]. The stability region is a region surrounded by the boundary line in the figure, and the vertical and the horizontal axes. XCP operates stably only when XCP control parameters (α, β) lie in the stability region. Note that in Fig. 2.3, boundary lines for (τ_1, τ_2) = (10, 10) and (20, 20) are almost identical.

In the following simulations, all XCP flows are activated simultaneously at $t = 0$. Note that staggered activation of XCP flows, instead of simultaneous activation, does not affect the stability of the XCP protocol if the activation interval of consecutive XCP flows is larger than the average round-trip time.

Figure 2.3 indicates that the stability region in a heterogeneous case (i.e., when two-way propagation delays τ_1 and τ_2 are different) is larger than that in the homogeneous case (i.e., when two-way propagation delays τ_1 and τ_2 are identical). This phenomenon can be explained by desynchronization of XCP flows with different propagation delays; i.e., when XCP flows have different propagation delays, variation in the transfer rate of an XCP flow is likely to be canceled by those of other XCP flows. Moreover, Fig. 2.3 indicates that in homogeneous cases (i.e., when two-way propagation delays of all XCP flows are identical) the stability region is independent of two-way propagation delays τ_1 and τ_2.

From these observations, we conclude that when XCP flows are heterogeneous, XCP operates more stably than the case when XCP flows are

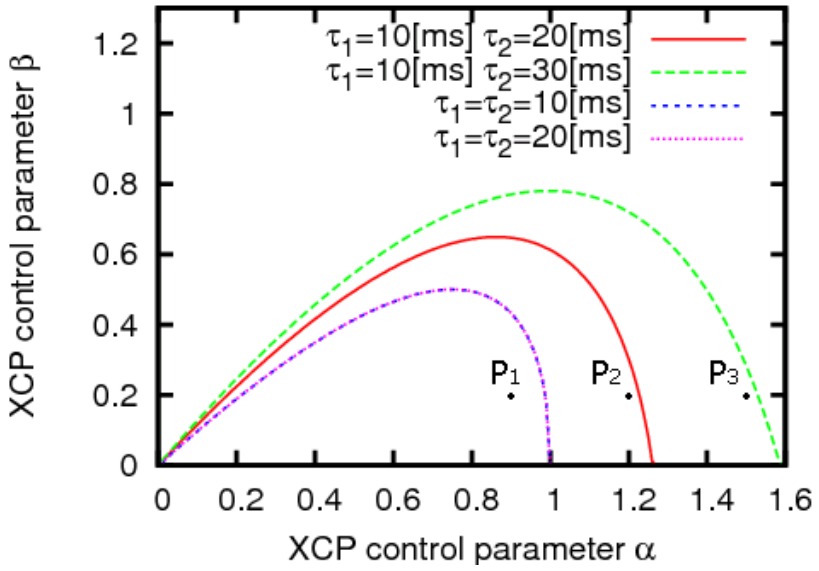


Figure 2.3: Stability region of XCP control parameters (α β) for different settings of two-way propagation delays (τ_1, τ_2); the stability region in a heterogeneous case is larger than that in the homogeneous case.

homogeneous.

We then examine the accuracy of our approximate analysis. We choose three settings of control parameters (α, β) : P_1 = (0.9, 0.2), P_2 = (1.2, 0.2), and P_3 = (1.5, 0.2) as shown in Fig. 2.3. For each control parameters setting, we perform simulation using ns-2 simulator for the same topology with Fig. 2.2. These control parameters settings are chosen to examine validity of our approximate analysis. For instance, if our stability analysis is valid, simulation results with P_2 should be stable for (τ_1, τ_2) = (10, 20), (10, 30) and unstable for (τ_1, τ_2) = (10, 10), (20, 20). In all simulations, all XCP flows are activated at 0 [s], the initial window size is set to 1 [packet], and the network topology shown in Fig. 2.2 is used.

Simulation results for control parameters settings P_1, P_2, and P_3 are

shown in Figs. 2.4 through 2.6, respectively. These figures show evolution of the window size of an XCP flow in flow class 1 for different settings of two-way propagation delays: i.e., (τ_1, τ_2) = (10, 10), (20, 20), (10, 20), and (10, 30). For each simulation result, stability of XCP is determined using a simple criterion — whether the window size of an XCP flow at the end of simulation is within $\pm$ 5% of its average value.

These simulation results are in agreement with our analytic results. For instance, the stability region shown in Fig. 2.3 indicates that, with the control parameters of $P_2 = (1.2, 0.2)$, the operation of XCP flows is unstable for (τ_1, τ_2) = (10, 10) and (20, 20), and is stable for (τ_1, τ_2) = (10, 20) and (10, 30). Simulation results shown in Fig. 2.5 clearly show that the operation of XCP flows is unstable for (τ_1, τ_2) = (10, 10) and (20, 20). Similar tendency can be observed for other control parameters P_1 = (0.9, 0.2) and P_3 = (1.5, 0.2). From these observations, we confirm validity of our approximate analysis.

Figures 2.4 through 2.6 include simulation results for (τ_1, τ_2) = (10, 20) with a large output link bandwidth $C = 800$ [Mbit/s]. One can find from these figures that stability of the XCP protocol is not affected by the output link bandwidth. As we have discussed in Section 2.4.1, these results coincide with our finding; i.e., the independence of the output link bandwidth C from stability of the XCP protocol.

2.4.3 Effect of variation in propagation delays

Next, the effect of variation in propagation delays of XCP flows on stability of the XCP protocol is investigated. In this chapter, CV(Coefficient of Variation) is used to measure variation in propagation delays, which is defined

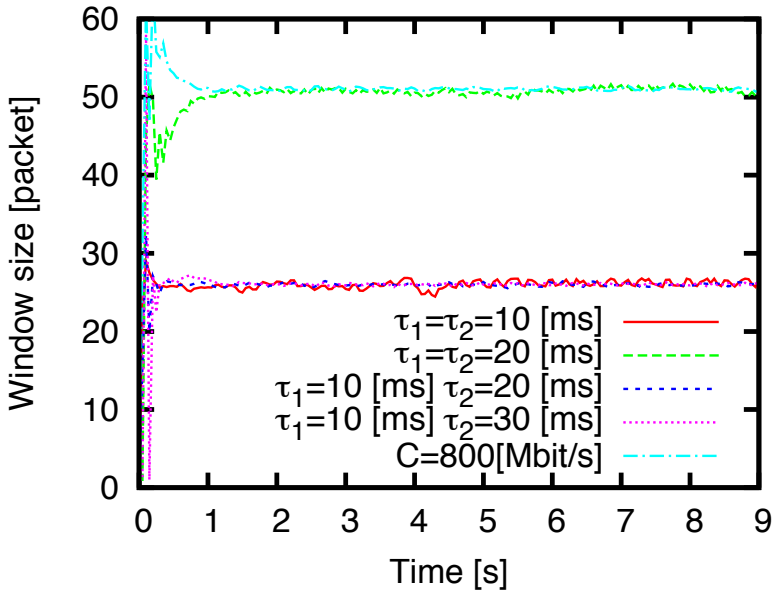


Figure 2.4: Simulation results with the control parameters of $P_1 = (0.9, 0.2)$; the window size of an XCP flow in flow class 1 is stable regardless of the two-way propagation delays (τ_1, τ_2).

by

$$CV = \frac{1}{d}\sqrt{\frac{\sum_{i=1}^{M}(\tau_i - d)^2}{\sum_{i=1}^{M} N_i}}. \tag{2.22}$$

Figure 2.7 shows stability regions of XCP control parameters (α β) for different numbers of XCP flows in each flow class: i.e., (N_1, N_2) = (5, 15), (10, 10), and (15, 5). In this figure, propagation delays of XCP flows, τ_1 and τ_2, are set to 10 [ms] and 20 [ms], respectively.

Figure 2.7 shows that the stability region is smallest when (N_1, N_2) = (15, 5); i.e., when the number of XCP flows in flow class 1 is larger than that in flow class 2. This phenomenon can be explained as follows. As explained in Section 2.1, both the efficiency controller and the fairness controller are invoked every average round-trip time of XCP flows. When there exist

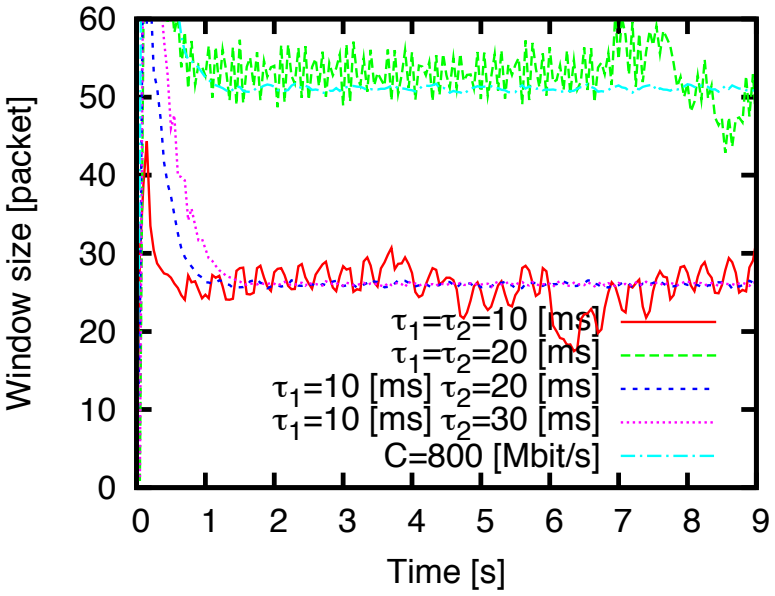


Figure 2.5: Simulation results with the control parameters of $P_2 = (1.2, 0.2)$; the window size of an XCP flow in flow class 1 is unstable for (τ_1, τ_2) = (10, 10), (20, 20).

many XCP flows with a small propagation delay, the average round-trip time estimated by the XCP router tends to be small. Since the XCP router invokes its controllers every the average round-trip time, these controllers are invoked quite frequently. This results in aggressive control of the XCP router, leading unstable operation.

From these observations, we conclude that when variation in propagation delays of XCP flows is large, operation of XCP becomes unstable.

We then investigate the effect of heterogeneity in XCP flows on stability of the XCP protocol. Figure 2.8 shows the maximum modulus of eigenvalues of the state transition matrix $\mathbf{B}$ for different numbers of XCP flows in flow class 1, N_1. In this figure, the number of XCP flows in flow class 2, N_2, is fixed at 1, and the propagation delay of XCP flows in flow class 1, τ_1, is at 10 [ms]. Note that control parameters (α, β) are set to their recommended

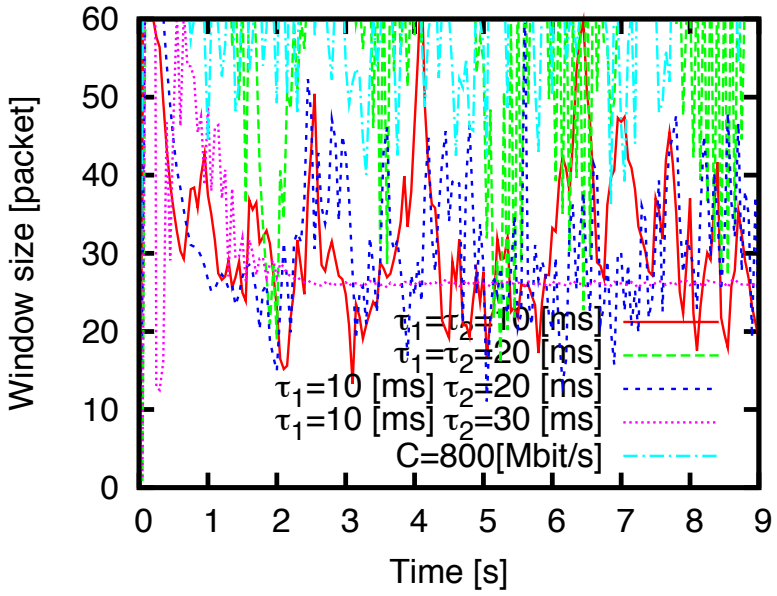


Figure 2.6: Simulation results with the control parameters of $P_3 = (1.5, 0.2)$; the window size of an XCP flow in flow class 1 is stable only when the two-way propagation delays (τ_1, τ_2) are set to (10, 30).

values, (0.4, 0.226) [24].

Figure 2.8 shows, for example, the operation of XCP becomes unstable when the number of XCP flows in flow class 1, N_1, reaches around 100 (i.e., the maximum modulus becomes larger than 1.0) for $\tau_2 = 200$ [ms].

Finally, through simulation experiments, we confirm the validity of our stability analysis and also investigate how XCP operates unstably when the heterogeneity of XCP flows is too large.

Figures 2.9 through 2.11 show evolution of the window size of an XCP flow in each flow class, and the evolution of the queue length of the XCP router. In these figures, the number of XCP flows in each flow class, (N_1, N_2), are set to (99, 1). Also, the two-way propagation delay of each flow class, (τ_1, τ_2), are to either (10, 100) or (10, 200). Fig. 2.9 is the simulation result of XCP flows with a small propagation delay. Fig. 2.10 is the simulation

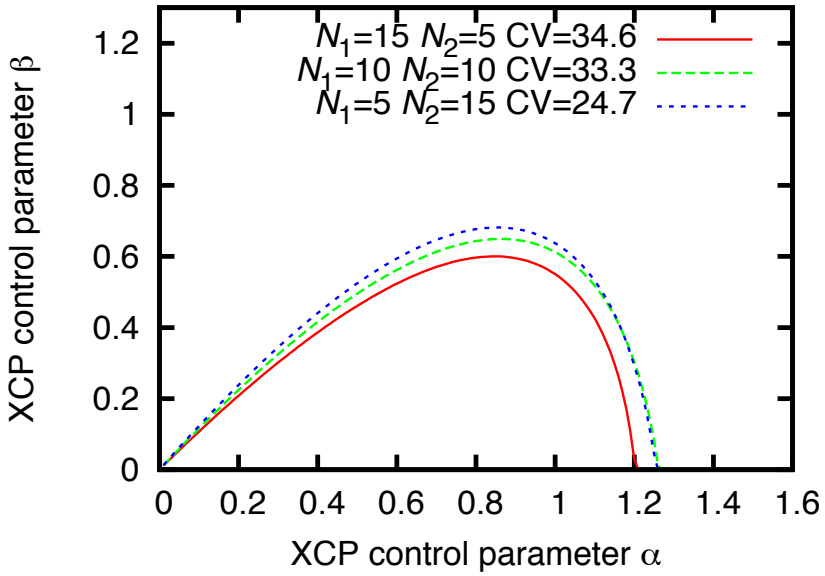


Figure 2.7: Stability region of XCP control parameters (α β) for different numbers of XCP flows in each flow class (N_1, N_2); the stability region is smallest when (N_1, N_2) = (15, 5).

result of XCP flows with a long propagation delay.

Figures 2.10 and 2.11 indicate that, when the two-way propagation delay τ_2 of the XCP flow in flow class 2 is 200 [ms], its window size and the queue length of the XCP router show oscillatory behavior, leading low throughput. Namely, when the variation in propagation delays of XCP flows is large, the window size of the XCP flow and the queue length of the XCP router become unstable, which shows the validly of our stability analysis.

2.5 Summary

In this chapter, we have analyzed the stability of XCP in a network with heterogeneous XCP flows (i.e., XCP flows with different propagation delays).

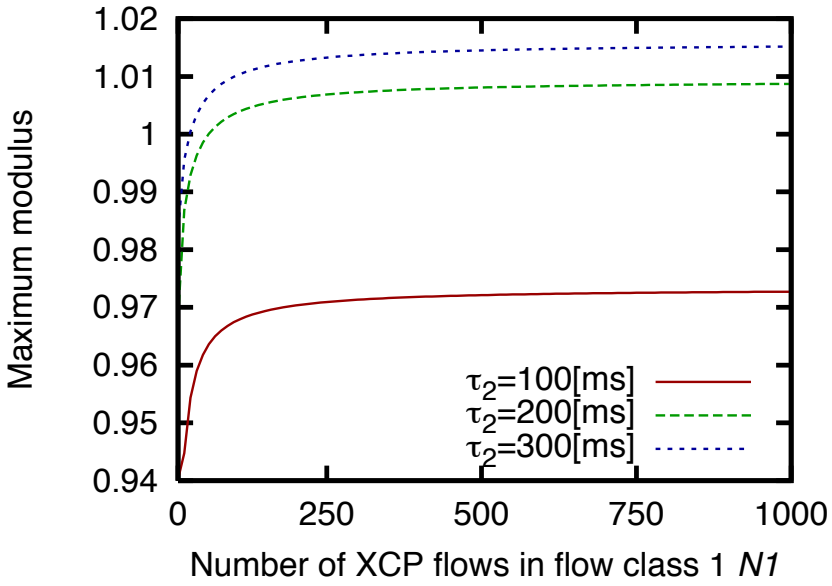


Figure 2.8: Maximum modulus of eigenvalues of the state transition matrix $\mathbf{B}$ for $\tau_1 = 10$ [ms], $N_2 = 1$, and $(\alpha,\beta) = (0.4,0.226)$; the operation of XCP becomes unstable when the number of XCP flows in flow class 1, N_1, reaches around 100 for $\tau_2 = 200$ [ms].

Through several numerical examples and simulation results, we have investigated the effect of system parameters and XCP control parameters on stability of the XCP protocol. Our findings include: (1) when XCP flows are heterogeneous, XCP operates more stably than the case when XCP flows are homogeneous, (2) conversely, when variation in propagation delays of XCP flows is large, operation of XCP becomes unstable, and (3) the output link bandwidth of an XCP router is independent of stability of the XCP protocol.

As future work, we are planning to analyze the transient performance of XCP utilizing our fluid model of XCP. In addition, we are planning to derive the optimal configuration of XCP control parameters, which maximizes the performance of XCP, based on our stability analysis and transient

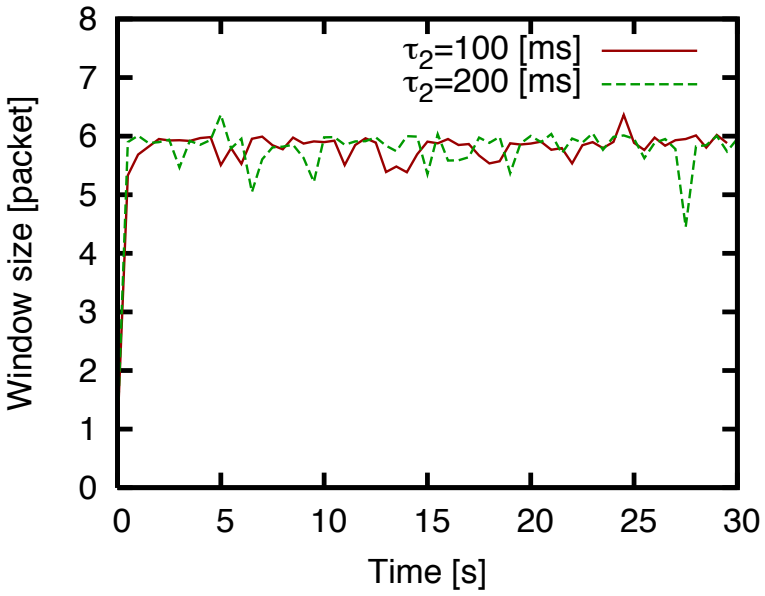


Figure 2.9: Evolution of the window size of an XCP flow in flow class 1 for τ_1 = 10 [ms] and (α, β) = (0.4, 0.226); XCP flows in flow class 1 are stable regardless of the two-way propagation delay τ_2 of flow class 2. Maximum throughput are 4.00 [Mbit/s] for τ_2 = 100 [ms] and 200 [ms].

performance analysis. We are planning to clarify the stability of XCP in a network with frequent XCP flows.

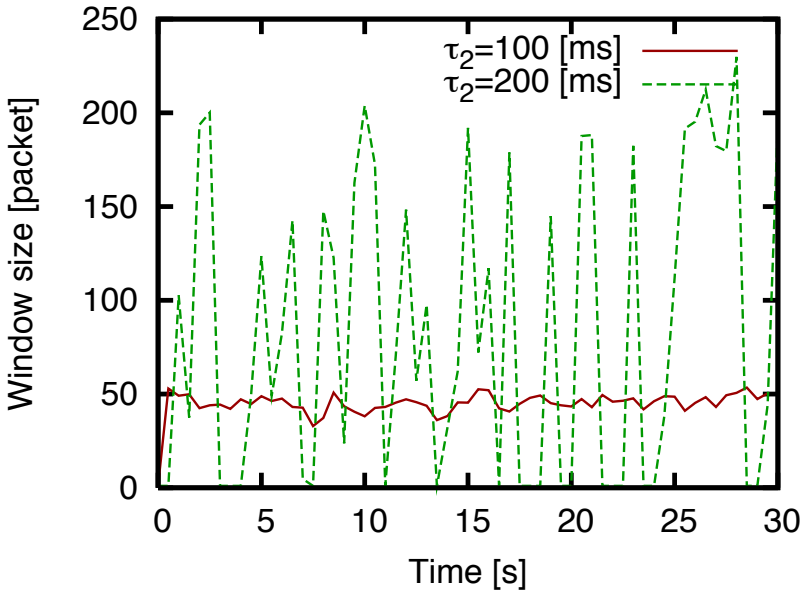


Figure 2.10: Evolution of the window size of an XCP flow in flow class 2 for τ_1 = 10 [ms] and (α, β) = (0.4, 0.226); XCP flows in flow class 2 become unstable when the two-way propagation delay τ_2 of flow class 2 is large. Maximum throughput are 4.00 [Mbit/s] for τ_2 = 100 [ms] and 200 [ms].

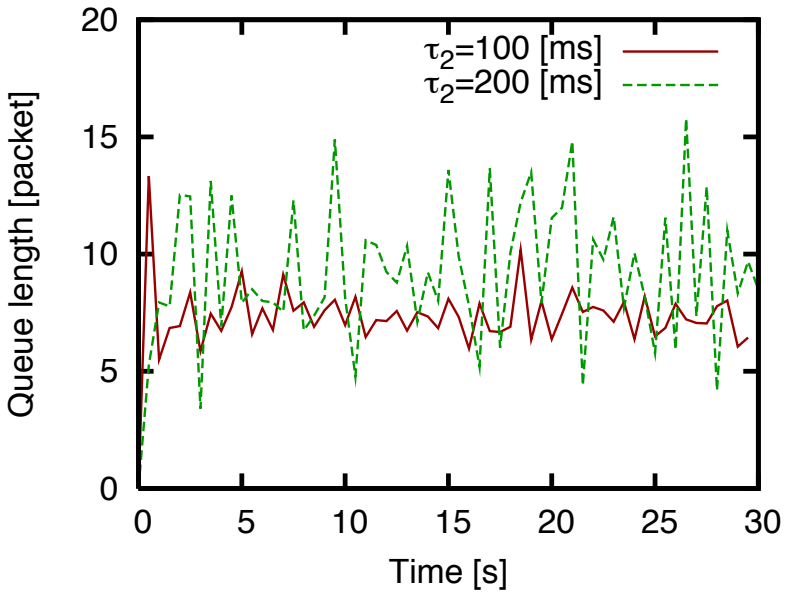


Figure 2.11: Evolution of the queue length of the XCP router for $\tau_1 = 10$ [ms] and (α, β) = (0.4, 0.226); The XCP router becomes unstable when the two-way propagation delay τ_2 of flow class 2 is large.

Chapter 3

Improving Robustness of XCP (eXplicit Control Protocol) for Dynamic Traffic

3.1 XCP (eXplicit Control Protocol)

3.1.1 Overview

In this section, we briefly summarize the operation algorithm of XCP. Refer to [24, 56] for details of the XCP protocol.

- XCP end host algorithm

  At the time of packet transmission, an XCP sender stores its estimated round-trip time, its current window size, and the initial value of the feedback value (i.e., the amount of window size increase requested by the XCP sender) in the congestion header of the packet.

  An XCP receiver receives the packet, and sends an ACK packet back

to the XCP sender. The XCP receiver simply copies the congestion header of the received packet to the congestion header of the ACK packet.

When an XCP sender receives the ACK packet, the XCP sender updates its window size, and re-calculates the estimated round-trip time. The window size is updated based on the amount of window size increase/decrease notified by XCP routers. More specifically, the window size is set to the sum of the current window size and the window size increase/decrease. The round-trip time is re-calculated using the algorithm similar to that of TCP Reno [57].

- XCP router algorithm

  The control mechanism of an XCP router is composed of the *efficiency controller*, which tries to maximize the link bandwidth utilization, and the *fairness controller*, which tries to realize fairness among competing XCP flows. The efficiency controller and the fairness controller are invoked every average round-trip time of all XCP flows.

  The efficiency controller estimates the amount of total rate increase/decrease for all XCP flows. The fairness controller then calculates the amount of rate increase/decrease for each XCP flow. An XCP router calculates the *feedback value* based on the amount of rate increase/decrease calculated by the fairness controller and information stored in the congestion header of arriving packets. In what follows, we briefly explain algorithms of the efficiency controller and the fairness controller.

  The efficiency controller calculates the *aggregate feedback value* ϕ, which is the amount of total rate increase/decrease for all XCP flows, from

the packet arrival rate at the XCP router and the queue length as

$$\phi = \alpha\, d\, (C - A) - \beta\, Q, \tag{3.1}$$

where d is the average round-trip time of XCP flows accommodated in the XCP router, C is the output link bandwidth of the XCP router, A is the packet arrival rate at the XCP router, Q is the minimum queue length observed during the average round-trip time, and α and β are control parameters. The efficiency controller controls so that: (1) the link bandwidth is fully utilized, and (2) the number of packets in the buffer becomes zero.

The fairness controller distributes the aggregate feedback value ϕ to all XCP flows based on an AIMD (Additive Increase and Multiplicative Decrease) discipline. Namely, if ϕ is positive, the fairness controller evenly allocates ϕ to all XCP flows so that the increase in throughput of all XCP flows is the same. Otherwise, the fairness controller unevenly allocates ϕ to all XCP flows so that the decrease in throughput of an XCP flow becomes proportional to its current throughput.

3.1.2 XCP problems for dynamic traffic

We investigate the implications of both XCP traffic dynamics and non-XCP (e.g., TCP and UDP) traffic dynamics on the XCP protocol.

First, we consider XCP traffic dynamics. The time evolution of the bottleneck link utilization (i.e., link utilization measured for every 10 [ms]) for the same simulation model with [24] is shown in Fig. 3.1. The amount of XCP traffic is fluctuated by changing the number of active XCP flows in a network similarly to [24]. In this simulation, 10 XCP flows are activated at

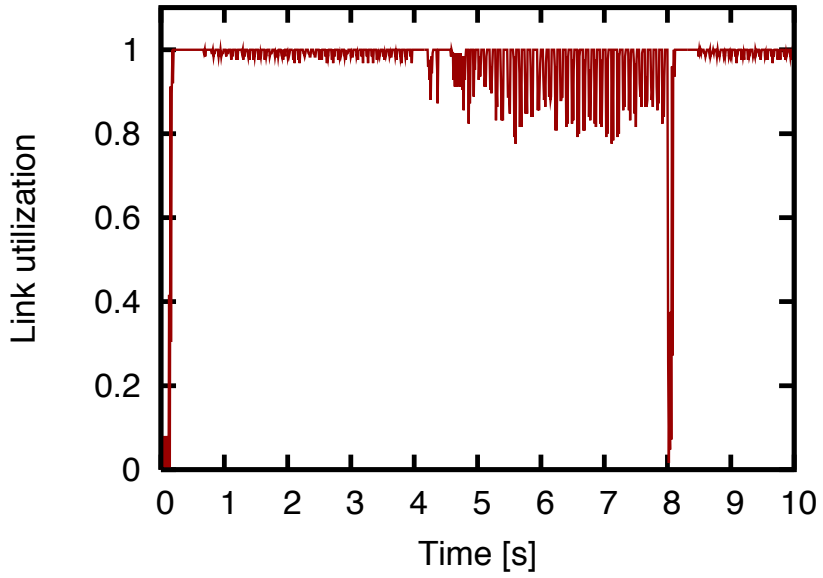


Figure 3.1: Time evolution of bottleneck link utilization when changing the number of active XCP flows at t = 4 and 8; the bottleneck link utilization degrades significantly when XCP flows terminate their transfers at $t = 8$.

$t = 0$, and 100 XCP flows are activated at $t = 4$. Also, 100 XCP flows are deactivated at $t = 8$.

Figure 3.1 shows that the bottleneck link utilization degrades significantly when XCP flows terminate their transfers at $t = 8$. This is because XCP controls the queue length of a router to become zero. Since the router's buffer is almost always empty, the bottleneck link utilization will degrade even when the amount of incoming XCP traffic slightly decreases. In a network with rapid XCP traffic dynamics like realistic XCP traffic dynamics, this phenomenon leads low bottleneck link utilization. As we will see in Section 3.4.1, this phenomenon becomes problematic for realistic XCP traffic dynamics.

Second, we consider non-XCP traffic dynamics. Simulations are per-

formed using the same network topology with that in Section 3.3. The time evolution of the number of packets in the XCP router's buffer (i.e., the queue length) when transmitting UDP traffic with the average transfer rate of 1.25, 2.5, and 3.75 [packet/ms] is shown in Fig. 3.2.

Figure 3.2 shows that the queue length of the XCP router becomes large when the amount of non-XCP traffic increases. This is because the XCP router performs its control by assuming that the available bandwidth of its output links is known (see Eq. (3.1)). If the amount of non-XCP traffic increases, the XCP router will be overloaded and many packets will be queued at the buffer. In a network with rapid non-XCP traffic dynamics like realistic TCP traffic dynamics, this phenomenon leads unstable XCP control. As we will see in Section 3.4.2, this phenomenon also becomes problematic for realistic non-XCP traffic dynamics.

From these observations, we conclude that XCP has the following problems: (1) the bottleneck link utilization is lowered against XCP traffic dynamics, and (2) operation of XCP becomes unstable in a network with both XCP and non-XCP traffic.

3.2 XCP-IR (XCP with Increased Robustness)

In this section, we explain the basic ideas for improving the robustness of XCP for traffic dynamics, and describe the operation algorithm of XCP-IR.

3.2.1 Basic ideas

First, how the degradation of the link utilization caused by XCP traffic dynamics can be prevented? To prevent degradation of the link utilization caused by XCP traffic dynamics, a router's buffer can be utilized ef-

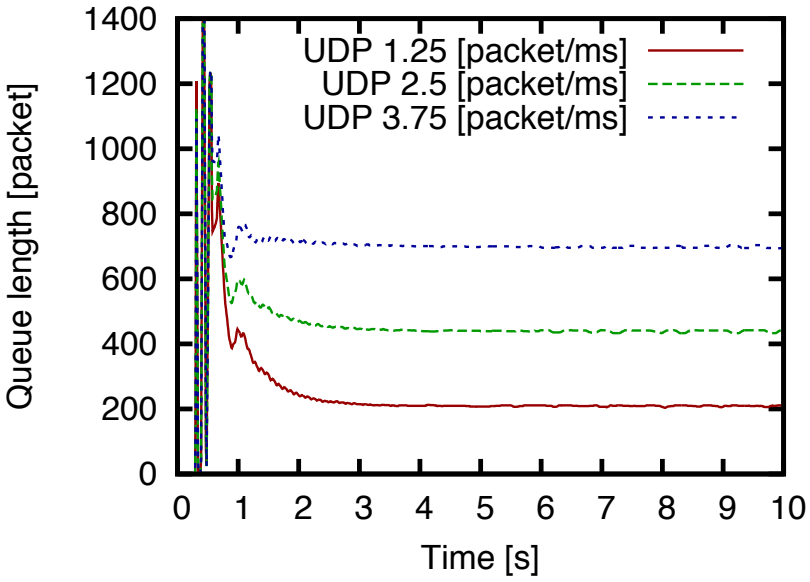


Figure 3.2: Time evolution of queue length at XCP router when transmitting non-XCP (UDP) traffic; the queue length of the XCP router becomes large as the amount of non-XCP traffic increases.

fectively. Namely, XCP traffic dynamics can be absorbed at the router's buffer. If a certain amount of packets are always stored in the XCP router's buffer, degradation of the link utilization caused by XCP traffic dynamics can be prevented. In the literature, congestion controls with storing a certain amount of packets in the bottleneck router's buffer have been proposed [17, 58, 59]. In these congestion controls, storing a certain amount of packets in the bottleneck router's buffer is effective for preventing the degradation of the link utilization.

Second, how instability of the XCP router caused by the increase in non-XCP traffic can be prevented? Instability of the XCP router control caused by the increase in non-XCP traffic can be prevented if the XCP router estimates the available bandwidth to XCP traffic correctly. For this purpose, an XCP router should measure the departure rate of non-XCP traffic. The

XCP router estimates the available bandwidth to XCP traffic by subtracting the departure rate of non-XCP traffic from the physical link bandwidth. The XCP router then distributes the available bandwidth for XCP traffic to all XCP flows. Since the XCP router can correctly calculate the bandwidth assigned to each XCP flow, stable control can be realized regardless of non-XCP traffic dynamics.

These improvements are realizable only by changing the control algorithm of an XCP router. Namely, it is necessary to change neither an XCP sender nor a packet format. Hence, the burden of deploying XCP-IR into a real network is quite low.

3.2.2 XCP-IR algorithm

In what follows, we explain the algorithm of XCP-IR. With XCP-IR, only the method of calculating the aggregate feedback value ϕ (Eq. (3.1)) is different from that of XCP. XCP-IR calculates the aggregate feedback value ϕ as

$$\phi = \alpha\, d\,\{(C - D_N) - A\} - \beta\,(Q - Q_T), \tag{3.2}$$

where D_N is the departure rate of non-XCP traffic, and Q_T is the target value of a queue length.

Q_T is a control parameter (i.e., the target value of the queue length) introduced to prevent degradation of the link utilization caused by XCP traffic dynamics. Thus, Q_T packets are always stored in the XCP router's buffer. Thereby, degradation of the link utilization can be prevented even for XCP dynamic traffic. The increase in Q_T may cause side-effects(e.g., the increase in a queuing delay and a packet loss probability, and less responsiveness). We discuss these side-effects with Section 3.3 and 3.4 for details.

D_N is an internal variable introduced to prevent instability of the XCP control caused by the increase in non-XCP traffic. An XCP router calculates D_N as

$$D_N = \frac{T}{d}, \tag{3.3}$$

where T is the total amount of non-XCP packets departed at the XCP router during the control interval d of the XCP router. By changing the available bandwidth of XCP traffic from C to $C - D_N$, XCP-IR can absorb XCP router's temporary overload when the amount of non-XCP traffic increases.

Note that such a simple modification to the XCP router itself may cause a fairness issue between XCP and non-XCP traffic. Specifically, changing the available bandwidth of XCP traffic from C to $C - D_N$ implies that XCP traffic suffers lower priority than non-XCP traffic. However, priority control between XCP and non-XCP traffic can be easily realized by introducing one of service differentiation mechanisms at an XCP router. For instance, a multi-level RED mechanism [60] can be applied to the XCP router for realizing priority control between XCP and non-XCP (e.g., TCP and UDP) traffic. By configuring parameters of the weighted round-robin queue appropriately, fairness among XCP and non-XCP traffic can be realized. Note that a multi-level RED mechanism is for realizing fairness among traffic classes (i.e., XCP and non-XCP traffic) rather than among traffic flows (i.e., XCP and non-XCP flows).

In what follows, we briefly demonstrate that the problem of XCP for XCP and non-XCP traffic dynamics can be solved using XCP-IR.

Figure 3.3 shows the time evolution of the bottleneck link utilization when using XCP-IR in the same simulation model with that of Fig. 3.1. The

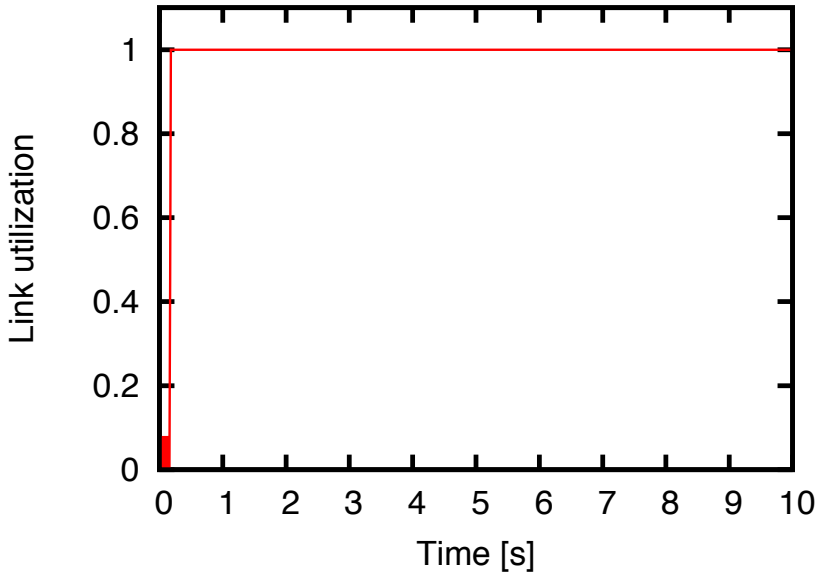


Figure 3.3: Time evolution of bottleneck link utilization when changing the number of active XCP flows at t = 4 and 8; XCP-IR completely prevents degradation of the bottleneck link utilization caused by XCP traffic dynamics.

target value Q_T of the queue length is set to 2,000 [packet]. By comparing Figs. 3.1 (XCP) and 3.3 (XCP-IR), one can find that XCP-IR completely prevents degradation of the bottleneck link utilization caused by XCP traffic dynamics.

Figure 3.4 shows the time evolution of the queue length of the XCP-IR router when transmitting the UDP traffic with the average transfer rate of 1.25, 2.5, and 3.75 [packet/ms]. To simply focus on the tolerance to non-XCP traffic dynamics, the target value of the queue length Q_T is set to 0 [packet]. By comparing Figs. 3.2 (XCP) and 3.4 (XCP-IR), one can find that XCP-IR succeeds to control the queue length regardless of the average transfer rate of UDP traffic.

In what follows, we discuss characteristics of XCP-IR. Specifically, we

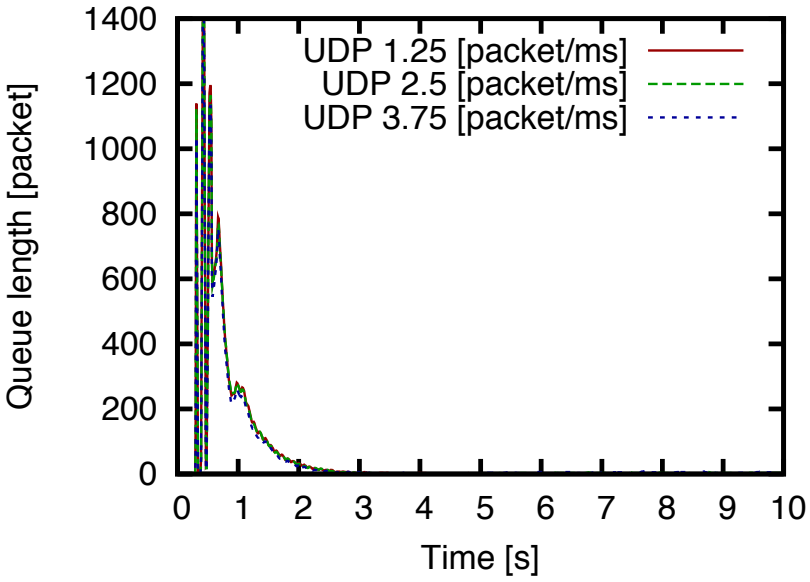


Figure 3.4: Time evolution of queue length at XCP-IR router when transmitting non-XCP (UDP) traffic; XCP-IR succeeds to control the queue length regardless of the average transfer rate of UDP traffic.

discuss how steady state and transient state performances are affected by changing the calculation method of the aggregate feedback ϕ from Eq. (3.1) to Eq. (3.2).

First, the round-trip times of XCP flows in XCP and XCP-IR are compared. The round-trip time of XCP-IR is larger than that of XCP. XCP-IR controls the queue length to Q_T whereas XCP controls to 0. In XCP-IR, the queuing delay of Q_T/C therefore occurs in the router's buffer. In other words, XCP-IR sacrifices the round-trip time for improving robustness. As we will discuss in Section 3.4, we believe such an increase in the round-trip time is acceptable for most applications.

Second, stability and transient state performance of XCP and XCP-IR are compared. We analyzed stability and transient state performance of

XCP and XCP-IR using the analytic approach in [33]. Although derivation and results are not presented due to space limitation, we found that the stability and the transient state performance of XCP-IR around the equilibrium point are same as those of XCP. Namely, our findings indicate that the stability and the transient state performance around the equilibrium point do not change, even if the calculation method of the aggregate feedback value is changed from Eq. (3.1) to Eq. (3.2). This implies that sensitivity of XCP-IR control parameters on stability and transient state performance is the same with that of XCP-IR.

3.3 Performance evaluation using synthetic traffic dynamics

In this section, we evaluate the performance of XCP-IR through extensive simulations using synthetic traffic dynamics. We investigate the performance of XCP-IR from the following viewpoints.

1. Stability and transient state performance
2. Robustness for XCP traffic dynamics
3. Robustness for non-XCP traffic dynamics

Figure 3.5 shows the network topology used in simulation. Except the existence of UDP traffic, we use the same simulation model with that in [24]. Multiple XCP flows and the single UDP flow share the single bottleneck link. For compact notation, in what follows, the bandwidth and the propagation delay of the bottleneck link are denoted by C and τ, respectively. Unless explicitly stated, parameter values shown in Tab. 3.1 are

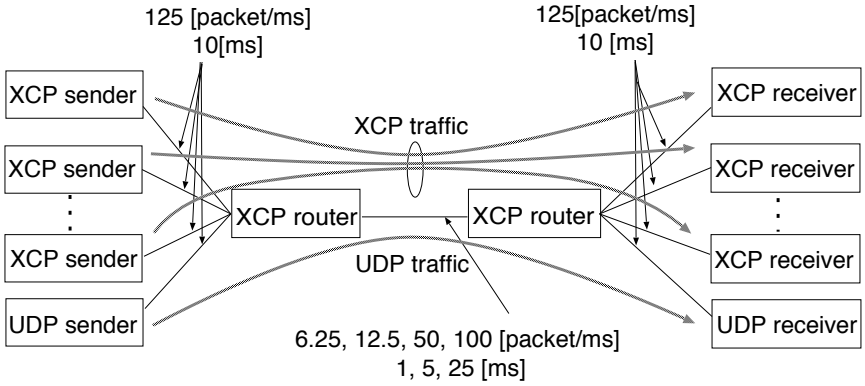


Figure 3.5: Network topology used in simulation; multiple XCP flows and the single UDP flow share the single bottleneck link.

Table 3.1: Parameters used in simulation

packet size	1,000 [byte]
socket buffer size	delay-bandwidth product
buffer size of XCP router	10,000 [packet]

used throughout our simulations. We used ns-2 version 2.28 with several modifications to implement XCP-IR.

3.3.1 Stability and transient state performance

Stability

In Section 3.2, the stability of XCP-IR around the equilibrium point was discussed. In the following simulations, we investigate the stability after initiating data transfers rather than the stability around the equilibrium point.

The stability regions of the control parameter (α, β) in XCP and XCP-IR are shown in Fig. 3.6. We performed a large number of simulations with different sets of control parameters (α, β). To focus on the effect of dynamic XCP traffic, 20 XCP flows started their transfers simultaneously (i.e., UDP

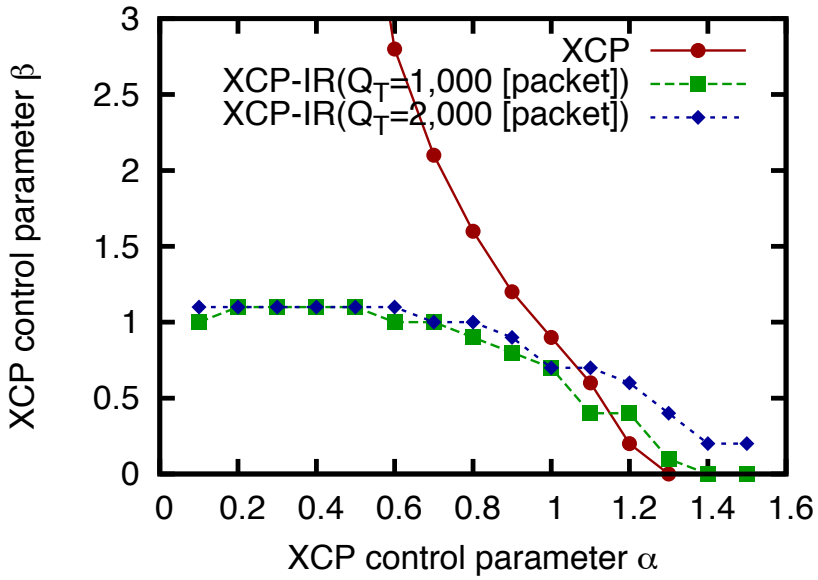


Figure 3.6: Stability region when XCP flow starts data transfers (XCP control is stable when (α, β) is under the boundary line); both XCP and XCP-IR operate stably with the recommended parameter configuration of (0.4, 0.226).

traffic is not generated). This figure means that operation of XCP or XCP-IR was stable, when a set of control parameters (α, β) was in the stability region (i.e., the region surrounded by the boundary line and XY-axes). We determined XCP or XCP-IR was stable when the window sizes of all flows were stabilized within $\pm$ 5% of their equilibrium values.

Figure 3.6 shows that the stability region of XCP-IR is smaller than that of XCP. Namely, the stability of XCP-IR is lower than that of XCP when XCP flows initiate their data transfers. However, with the recommended parameter configuration of (α, β) = (0.4, 0.226) [24], both XCP and XCP-IR operate stably. Hence, as long as the recommended parameter configuration is used, there should be no stability problem in XCP-IR.

Transient state performance

We also investigate the transient state performance after initiating data transfers rather than the transient state performance around the equilibrium point.

The settling time of the window size of an XCP flow while changing the target value of the queue length Q_T from 0 to 5,000 [packet] is shown in Fig. 3.7. The settling time of the window size is defined as the time for the window size to be stabilized within $\pm$ 5% of its equilibrium value (i.e., the window size in steady state). To focus on the effect of dynamic XCP traffic, 20 XCP flows start their transfers simultaneously in this simulation.

Figure 3.7 shows that the transient state performance of XCP-IR is always better than that of XCP unless the target queue length Q_T is too large. As the target queue length Q_T increases, the settling time of the window size once decreases. But it then increases as the target queue length becomes too large. This is because that an increase in the target queue length Q_T has both positive and negative effect on the transient state performance. Namely, increasing the target queue length makes XCP-IR more aggressive since the aggregate feedback value ϕ becomes large as Q_T increases. At the same time, increasing the target queue length makes XCP-IR less responsive since the round-trip time becomes large as Q_T increases. With an appropriate setting of the target queue length Q_T, the positive effect is much stronger than the negative effect, leading XCP-IR's better transient performance than XCP.

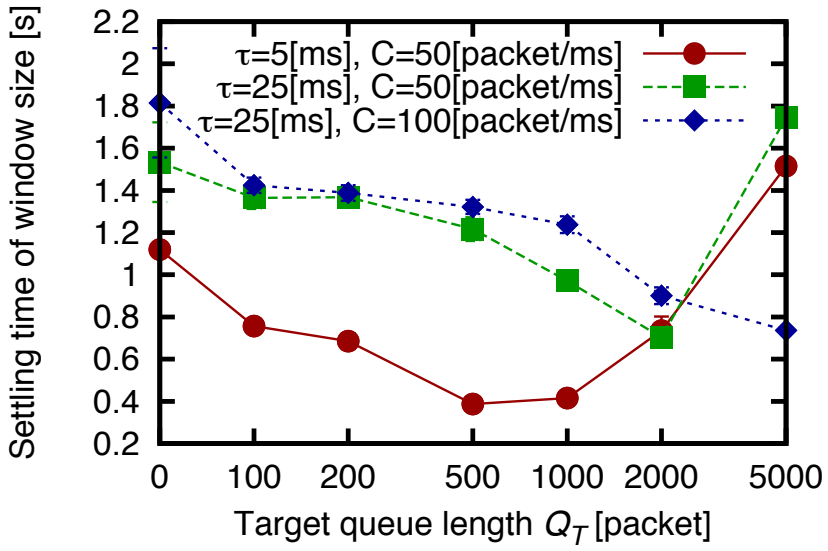


Figure 3.7: Settling time of window size when an XCP flow starts its data transfer; the settling time of the window size is a concave function for the target value of queue length Q_T.

3.3.2 Robustness for XCP traffic dynamics

For investigating robustness of XCP-IR, dynamic XCP traffic is synthetically generated by changing the number of active XCP flows as shown in Fig. 3.8. Namely, 10 XCP flows are activated every 1 [s] after $t = 0$, and 10 XCP flows are deactivated every 1 [s] after $t = 5$. To focus on the effect of dynamic XCP traffic, UDP traffic is not generated. The bottleneck link utilization when changing the target value of the queue length Q_T is shown in Fig. 3.9. Note that the result of XCP corresponds to the case with $Q_T = 0$ [packet].

Figure 3.9 shows that XCP-IR always shows higher bottleneck link utilization than XCP. This figure also shows the bottleneck link utilization increases as the target value of the queue length Q_T increases. It should be

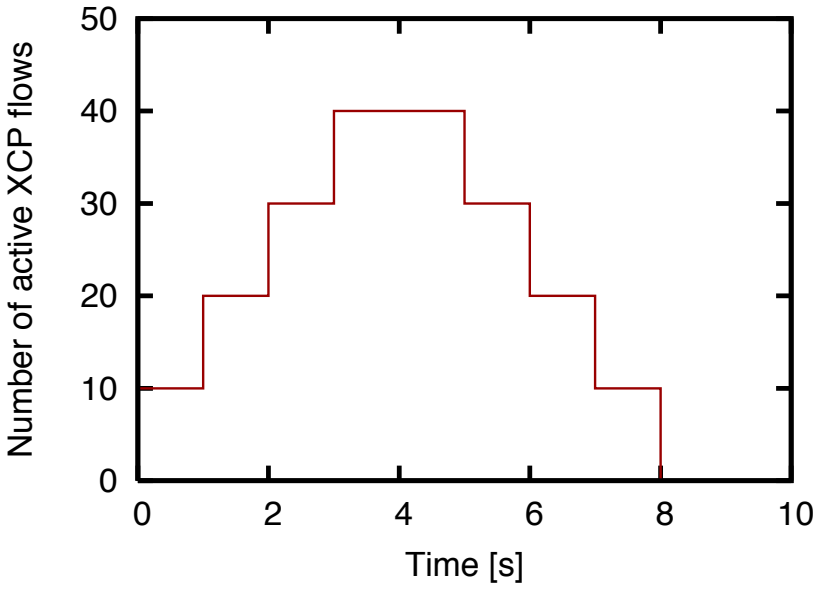


Figure 3.8: Synthetically generated XCP traffic dynamics; the number of active XCP flows are synthetically changed during simulation.

noted that all bottleneck link utilizations in Fig. 3.9 are convex for the target value Q_T. This implies an advantage of XCP-IR; i.e., a small increase in Q_T significantly improves the bottleneck link utilization. In practice, the target value of the queue length Q_T should be determined by taking account of trade-offs between robustness and responsiveness. For the link utilization, the improvement in XCP-IR is not so significant. However, in Section 5.1, we show that XCP-IR will significantly improve throughput.

Discussion on appropriate setting of Q_T

How the target value of the queue length Q_T should be configured to maximize the XCP-IR performance? When the number of active XCP flows decreases rapidly, at least the feedback delay of d_f is taken for the XCP router to notify XCP senders of the changed feedback value. The feedback delay

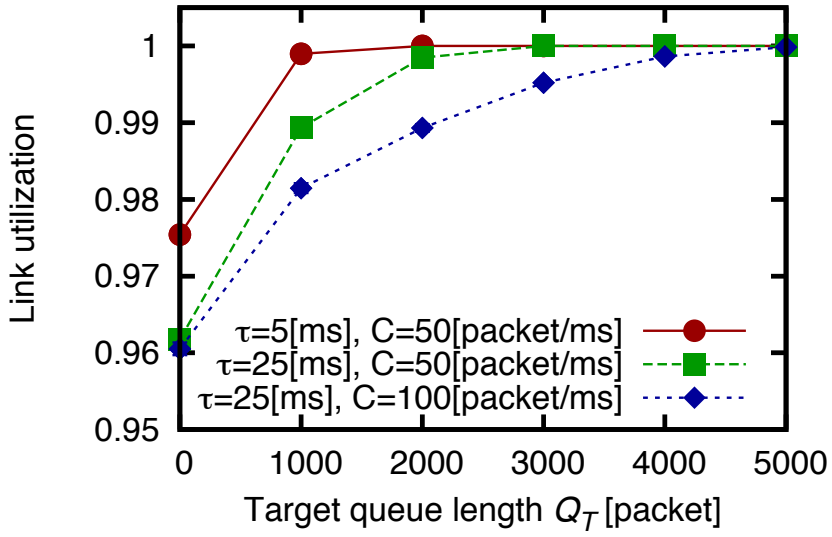


Figure 3.9: Bottleneck link utilization for a different target value Q_T with synthetic XCP traffic dynamics; XCP-IR always shows higher bottleneck link utilization than XCP.

d_f is a fraction of the round-trip time d; i.e., d_f can be obtained by subtracting the queuing delay from the round-trip time d. Let us assume that M of N XCP flows terminate their transfers. In this case, $d_f\, C\, M/N$ packets are drained from the buffer until XCP senders are notified of the re-calculated feedback value. To prevent degradation of the bottleneck link utilization, it is necessary to satisfy $d_f\, C\, M/N \leq Q_T$; i.e.,

$$Q_T \geq \frac{d_f\, C\, M}{N} \tag{3.4}$$

Figure 3.9 clearly shows the validity of Eq. (3.4); i.e., the bottleneck link utilization is almost 1.0 when the target value of the queue length satisfies Eq. (3.4). Namely, $Q_T \geq 1,250$ for τ = 5 [ms] and C = 50 [packet/ms], $Q_T \geq 2,250$ for τ = 25 [ms] and C = 50 [packet/ms], and $Q_T \geq 4,500$ for τ

= 25 [ms] and C = 100 [packet/ms].

We then focus on the time evolution of the bottleneck link utilization for τ = 25 [ms] and C = 50 [packet/ms]. In Fig 3.10, the time evolutions of the bottleneck link utilization in XCP and XCP-IR with $Q_T = 2,000$ and $Q_T = 2,600$ is plotted. The number of active XCP flows is synthetically changed as shown in Fig. 3.8. Figure 3.10 shows that degradation of the bottleneck link utilization can be prevented with Q_T larger than 2,250 [packet] (i.e., Q_T =2,600 [packet]).

Note that the bottleneck link utilization is stabilized quite rapidly in Fig. 3.10. This is because XCP can quickly respond to decrease in the number of active XCP flows. When the number of active XCP flows decreases rapidly, the queue length of the XCP router decreases, which results in smaller round-trip time to active XCP flows. Thus, XCP flows can quickly respond to congestion indication from the XCP router.

3.3.3 Robustness for non-XCP traffic dynamics

We first investigate the robustness of XCP-IR for synthetically generated dynamic UDP traffic. The average transfer rate of background UDP traffic is changed every second as shown in Fig. 3.11.

The time evolutions of the queue lengths with XCP and XCP-IR (Q_T = 0 and 600 [packet]) for τ = 25 [ms] and C = 50 [packet/ms] are shown in Fig. 3.12. A single XCP flow is activated at $t = 0$.

This figure shows that XCP fails to stabilize the queue length. This figure also shows that XCP-IR controls the queue length around the target value of the queue length Q_T. This clearly indicates that XCP-IR has robustness for UDP traffic dynamics.

3.4 Performance evaluation using realistic traffic dynamics

3.4.1 Robustness for realistic XCP traffic dynamics

For deploying XCP-IR in a real network, its performance should be evaluated under realistic conditions. We therefore investigate the robustness of XCP-IR for realistic XCP traffic dynamics.

Realistic dynamic XCP traffic is generated by aggregating a large number of XCP flows according to the mesurement results reported in [61]. Currently, majority of the Internet traffic is Web and P2P traffic. We generated realistic dynamic XCP traffic by aggregating a large number of XCP flows. Each XCP flow is randomly actived to match the measured distribution of flow activation intervals [61]. It is then deactivated when it finishes its file transfer, whose file size is randomly determined according to the measured distribution of file sizes [61]. More specifically, distributions of flow activation intervals of Web and P2P traffic are given by an exponential distribution with mean 0.015 and 14.98 [s], respectively. Distributions of file sizes of Web and P2P traffic are given by a Pareto distribution with mean $0.421C \times 10^{-3}$ and $0.121C$ [Mbyte], respectively. Note that file sizes are proportional to the bottleneck link bandwidth C. In our simulation scenario, the number of XCP flows carrying Web traffic is as appoximately 1,100 times as those carrying P2P traffic.

The bottleneck link utilization for different settings of C and τ are shown in Fig. 3.13. This figure shows that XCP-IR can achieve 97% link utilization with an adequate setting of the target queue legnth Q_T while XCP achieves only 86% link utilization. In this case, $Q_T = 600$ [packet] is suffi-

cient for well utilizing the bottleneck link bandwidth regardless of the bottleneck link bandwidth C and the propagation delay τ. Q_T can take a much smaller value while almost fully utilizing the bottleneck link bandwidth if the propagation delay is small (e.g., τ = 1 or 5 [ms]). Note that since this simulation scenario is very busrty, the bottleneck link utilization in XCP-IR is slightly less than 1.0. An example of realistic traffic dynamics (i.e., the evolution of the number of active XCP flows) generated by aggregating a large number of XCP flows is shown in Fig. 3.14. Note that the realistic traffic dynamics (Fig. 3.14) is much more dynamical than the synthetic traffic dynamics (Fig. 3.8).

The average round-trip delay of all XCP flows for different settings of C and τ are shown in Fig. 3.15. This figure shows that the average round-trip time of all XCP flows gradually increases as the target queue legnth Q_T increases. This is because XCP-IR controls so that Q_T packets are queued at the router's buffer to prevent degradation of the bottleneck link utilization. Note that since this simulation scenario is very busrty, the queue length with XCP-IR is smaller than Q_T. Therefore, the increase in the round-trip time of all XCP flows is smaller than Q_T/C. To prevent the significant increase in the round-trip delay of an XCP flow, Q_T should be set between 0 and 150 [packet] for LAN environment and between 0 and 600 [packet] for WAN environment.

Average throughputs of all Web flows and all P2P flows for different settings of C and τ are shown in Fig. 3.16 and 3.17. Figure 3.16 shows that the average throughput of all Web flows with XCP-IR is always higher than that of XCP unless the target queue length Q_T is too large. This is because as the target queue length Q_T increases, although increase in the link utilization become small, increase in the round-trip time do not become

small. Figure 3.17 shows that the average throughput of all P2P flows with XCP-IR is always higher than that of XCP. This is because there is almost no effect of the throughput on the increase in the round-trip time since file sizes of P2P traffic are large.

Due to space limitation, results for the packet loss probability in the bottleneck router for a different C and τ are not included. We have confirmed that a packet was not lost in the bottleneck router. If the configuration value of Q_T is sufficiently small as compared with the buffer size of the XCP router, increase in a packet loss probability can be prevented.

We then focus on the time evolution of the bottleneck link utilization for τ = 25 [ms] and C = 6.25 [packet/ms]. In Fig 3.18, the time evolutions of the bottleneck link utilization with XCP and XCP-IR for $Q_T = 150$ [packet] and $Q_T = 300$ [packet] is plotted. This figure shows that degradation of the bottleneck link utilization can be almost prevented with XCP-IR(Q_T = 300 [packet]).

From these observations, we conclude that XCP-IR has high robustness for different types of XCP traffic dynamics.

3.4.2 Robustness for realistic TCP traffic dynamics

We finally investigate the robustness of XCP-IR for realistic non-XCP traffic dynamics. Since the majority of the Internet traffic is carreid by TCP, we consider the impact of dynamic TCP traffic on the robustness of XCP-IR.

Similarly to Section 3.4.1, realistic dynamic TCP traffic is generated by aggregating a large number of TCP flows according to the mesurement results in [61]. Namely, TCP flows are randomly activated and their file sizes are randomly determined based on the measurement results in [61]. A sin-

gle XCP flows is activated at $t = 0$.

Time evolutions of queue lengths with XCP, XCP-IR (Q_T = 0 [packet]), and XCP-IR (Q_T = 600 [packet]) for τ = 25 [ms] and C = 6.25 [packet/ms] are shown in Fig. 3.19. This figure shows that the queue length of XCP drastically oscillates, whereas the queue length with XCP-IR is stabilized and minimized. Note that the queue length with XCP is unstable; the queue length is simply upper-bounded by the buffer size, and many packet are lost due to buffer overflows. These result indicate that the lack of robustness is quite costly for realistic dynamic traffic. The oscillatory behavior of the queue length with XCP in Fig. 3.19 is directly caused by the oscillatory behavior of XCP flows. Figure 3.20 shows the evolution of the window size of an XCP flow. As explained in Section 3.1, when the amount of non-XCP traffic increases, the XCP router is overloaded and many packets are queue at the buffer because the XCP router performs its control by incorrectly assuming that the available bandwidth to XCP traffic is known.

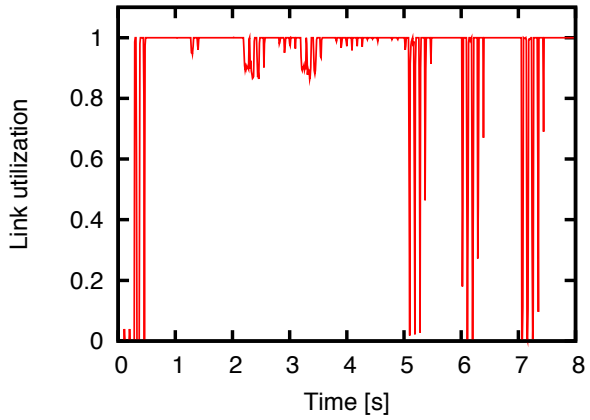
We then investigate how malicious XCP's lack of robustness is in terms of the bottleneck link utilization, the throughput of the XCP flow, and the round-trip time. The bottleneck link utilization, the throughput of the XCP flow, and the round-trip time of the XCP flow for different buffer sizes of the XCP router are shown in Figs. 3.21 through 3.23, respectively. Those figures show that the performance of XCP is significantly affected by the buffer size of the XCP router, but XCP still suffers either low bottleneck link utilization or large round-trip time. This can be explained as follows. If the buffer size of the XCP router is small, many packets are lost due to buffer overflows, leading low bottleneck link utilization and low throughput. On the contrary, if the buffer size of the XCP router is large, the queuing delay in the XCP router is large, leading a large round-trip time.

From these observations, we conclude that XCP-IR achieves good performance and high robustness for different types of non-XCP traffic dynamics.

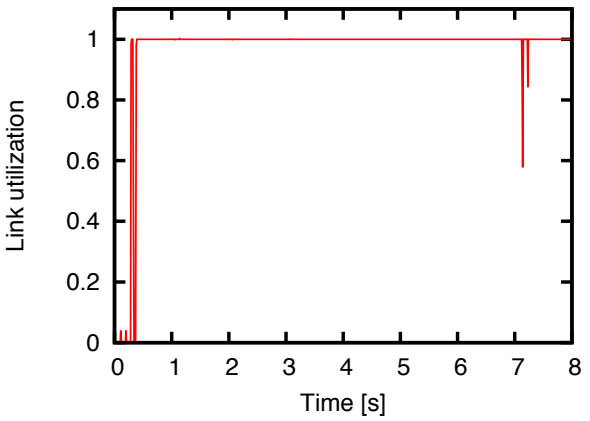
3.5 Summary

In this chapter, we have proposed XCP-IR (XCP with Increased Robustness) that operates efficiently even for dynamic traffic. XCP-IR prevents instability of the XCP control caused by non-XCP traffic dynamics while preventing degradation of the bottleneck-link utilization caused by XCP traffic dynamics. Through extensive simulation experiments, we have shown that XCP-IR operated efficiently even for dynamic traffic. In particular, we have shown that the throughput with XCP-IR is approximately 200% higher than that with XCP.

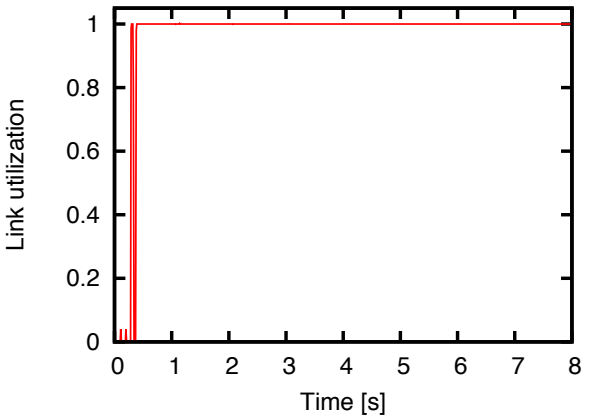
As future work, we are planning to examine the optimal control parameter configuration of XCP-IR for maximizing its performance in realistic environments.



(a) XCP



(b) XCP-IR($Q_T = 2,000$ [packet])



(c) XCP-IR($Q_T = 2,600$ [packet])

Figure 3.10: Time evolution of the bottleneck link utilization with XCP, XCP-IR($Q_T = 2,000$ [packet]), and XCP-IR($Q_T = 2,600$ [packet]) for τ = 25 [ms] and C = 50 [packet/ms]

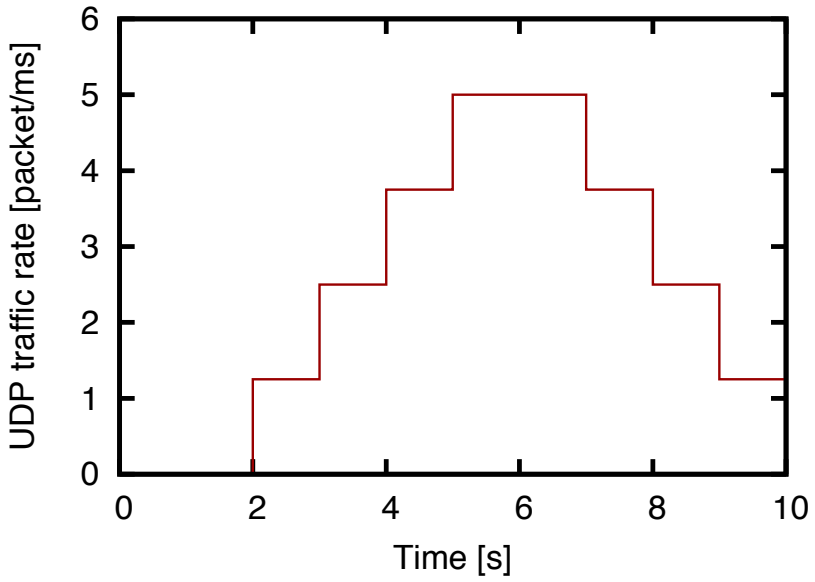


Figure 3.11: Synthetically generated UDP traffic dynamics; the average transfer rate of UDP traffic is changed during simulation.

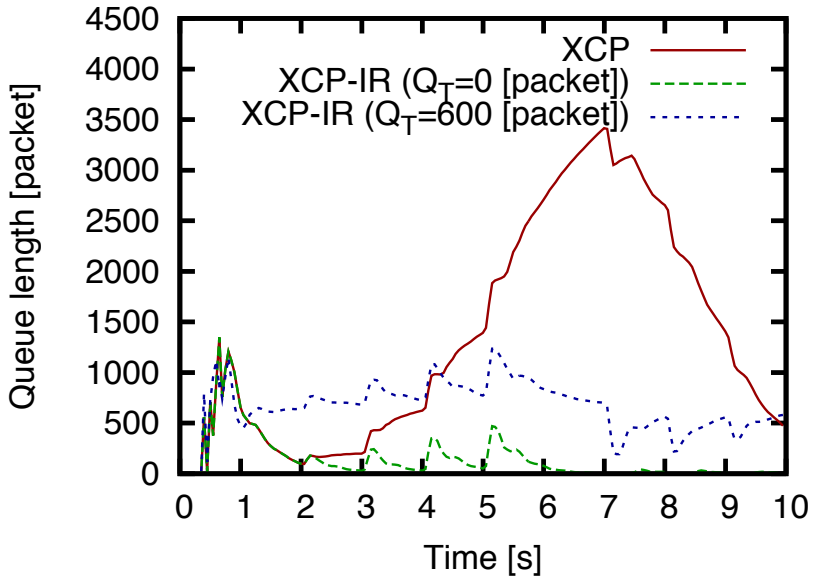


Figure 3.12: Time evolutions of queue lengths with XCP and XCP-IR (Q_T = 0 and 2,000 [packet]) for τ = 25 [ms] and C = 50 [packet/ms]; XCP-IR can control the queue length around the target value Q_T.

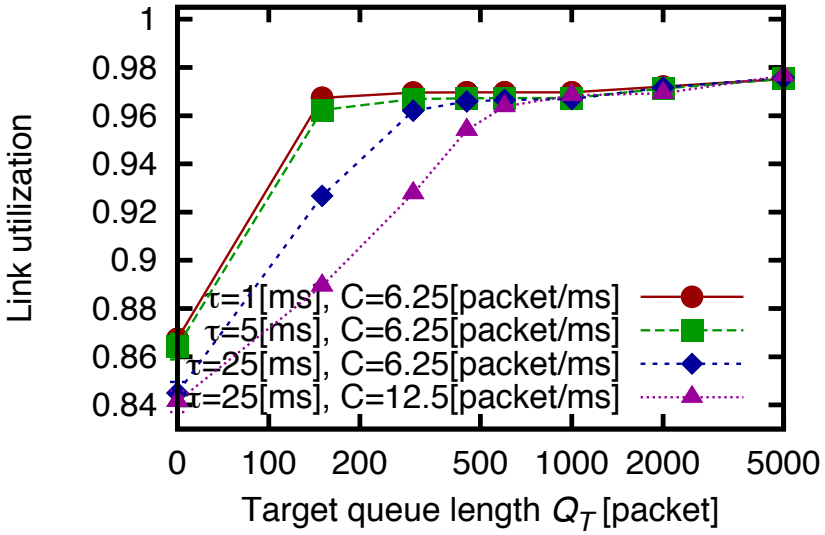


Figure 3.13: Bottleneck link utilization for a different C and τ; the bottleneck link utilization with XCP-IR is approximately 10% higher than that with XCP.

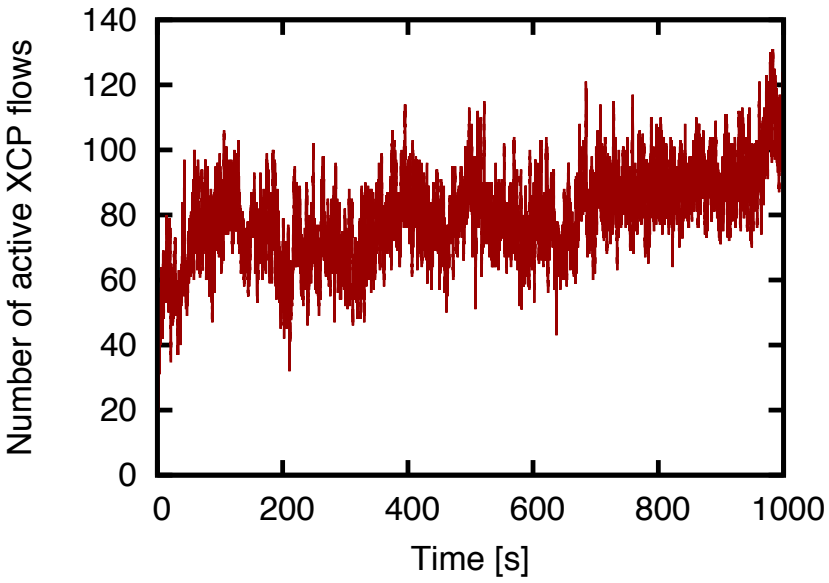


Figure 3.14: Time evolution of the number of active XCP flows for τ = 25 [ms] and C = 6.25 [packet/ms]

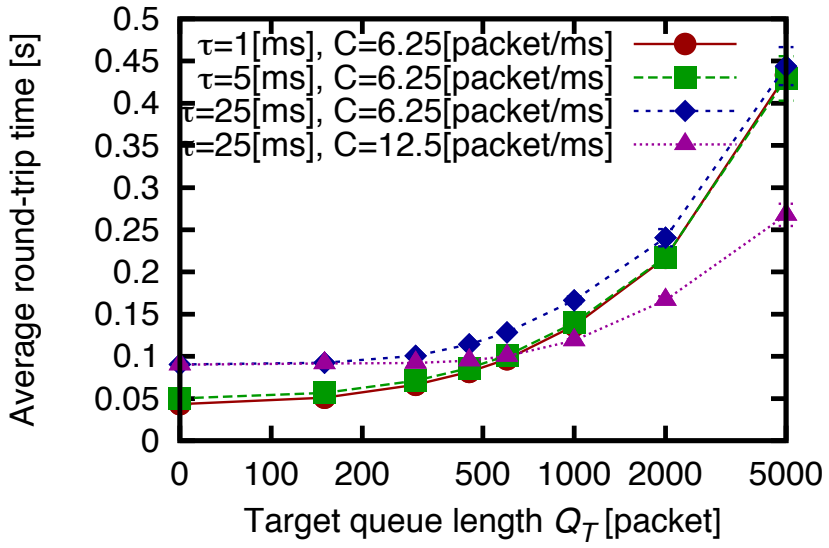


Figure 3.15: Average Round-trip time of all XCP flows for a different C and τ; the average round-trip time of all XCP flows gradually increases when Q_T increases.

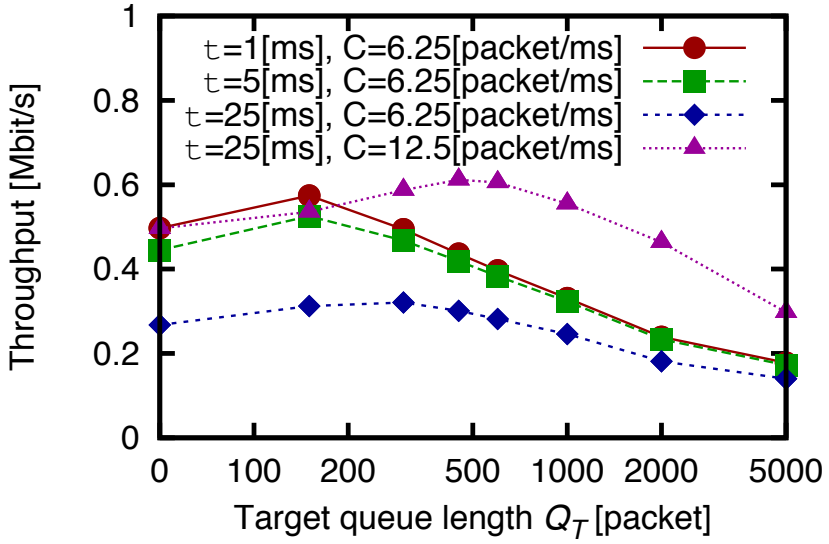


Figure 3.16: Average throughput of all Web flows for a different C and τ; the average throughput of all Web flows is a convex function for the target value of queue length Q_T.

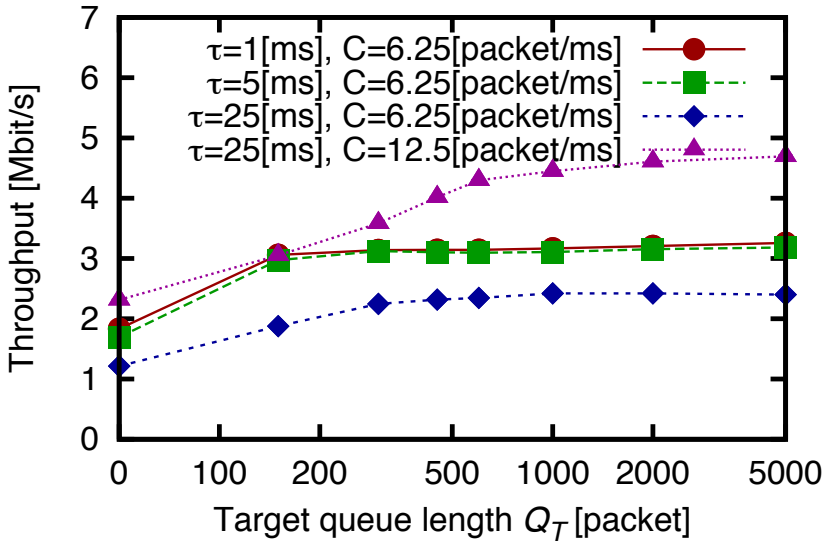
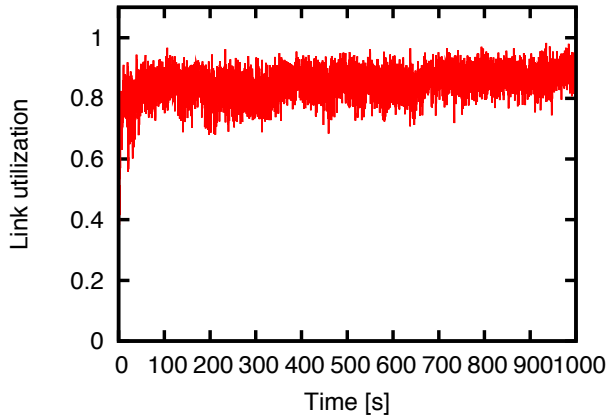
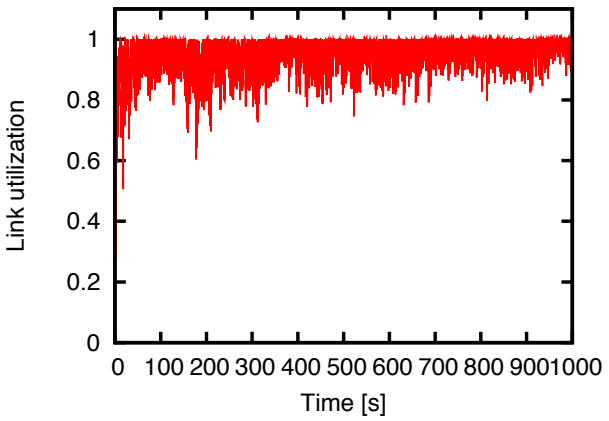


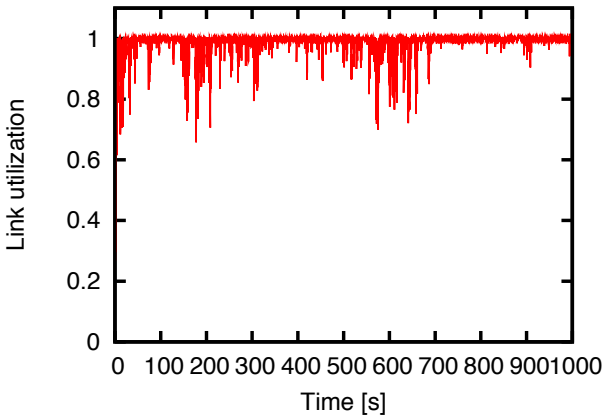
Figure 3.17: Average throughput of all P2P flows for a different C and τ; the average throughput of all P2P flows with XCP-IR is approximately 200% higher than that with XCP.



(a) XCP



(b) XCP-IR(Q_T = 150 [packet])



(c) XCP-IR(Q_T = 300 [packet])

Figure 3.18: Time evolution of the bottleneck link utilization with XCP, XCP-IR(Q_T = 150 [packet]), and XCP-IR(Q_T = 300 [packet]) for τ = 25 [ms] and C = 6.25 [packet/ms] when generating realistic XCP traffic

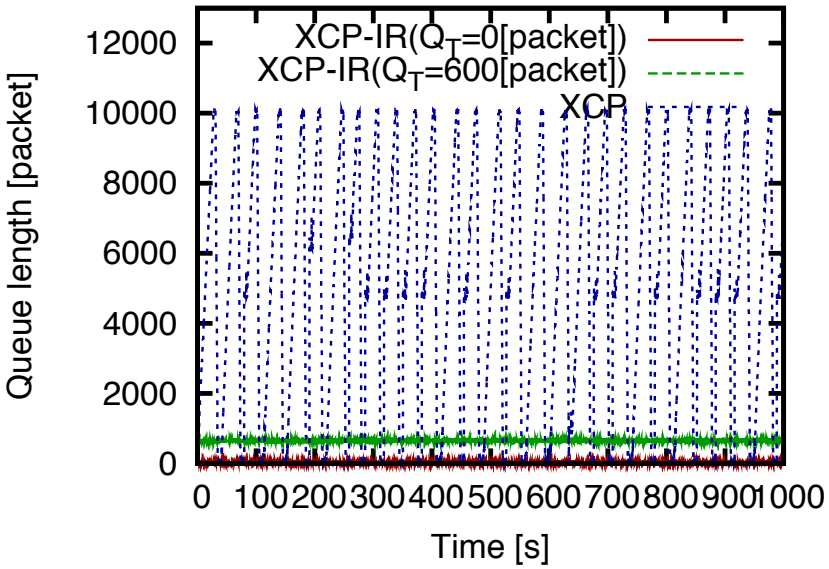


Figure 3.19: Time evolutions of queue lengths with XCP, XCP-IR (Q_T = 0 [packet]), and XCP-IR (Q_T = 600 [packet]) for τ = 25 [ms] and C = 6.25 [packet/ms]; the queue length of XCP drastically oscillates, whereas the queue length of XCP-IR is stabilized and minimized.

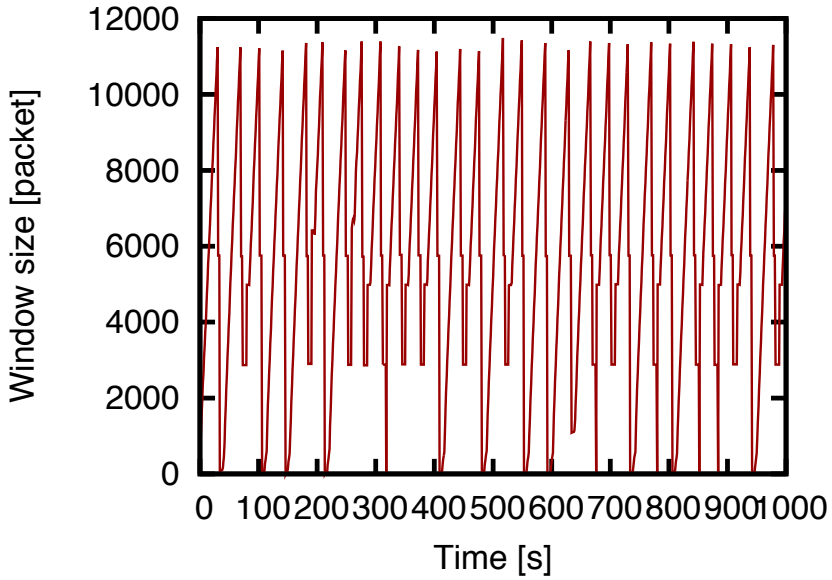


Figure 3.20: Time evolutions of the window size with XCP for τ = 25 [ms] and C = 6.25 [packet/ms]

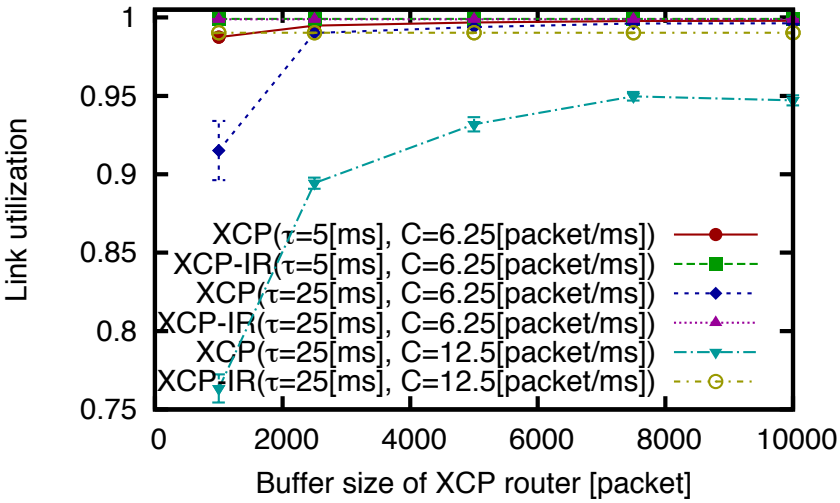


Figure 3.21: Bottleneck link utilization for a different buffer size of the XCP router; the bottleneck link utilization with XCP is low when the buffer size of the XCP router is small.

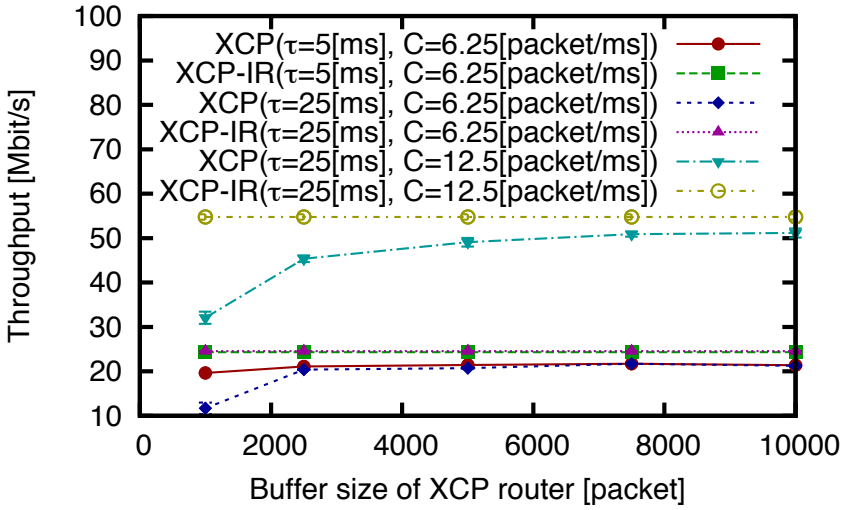


Figure 3.22: Throughput of the XCP flow for a different buffer size of the XCP router; the throughput of the XCP flow with XCP is low when the buffer size of the XCP router is small.

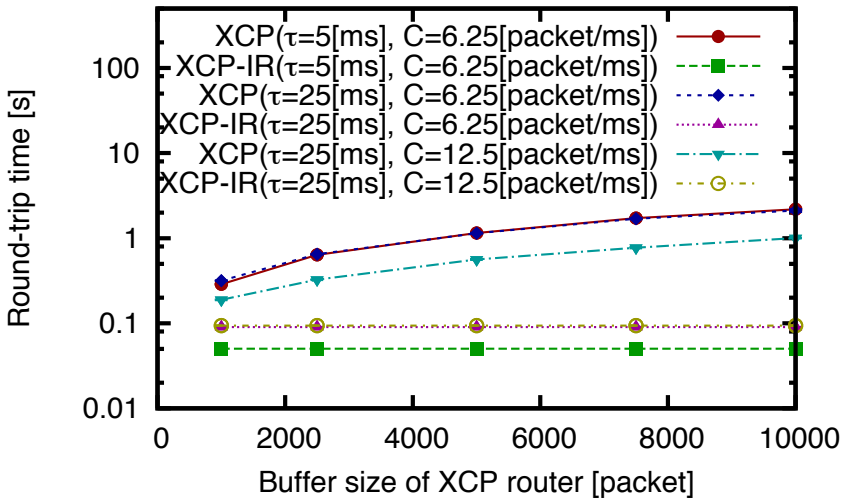


Figure 3.23: Round-trip time of the XCP flow for a different buffer size of the XCP router; the round-trip time of the XCP flow with XCP is very large when the buffer size of the XCP router is large.

Chapter 4

Design and Implementation of Flow-Level Simulator FSIM for Performance Evaluation of Large Scale Networks

4.1 Related Works

In [39], an approach for large-scale network simulation utilizing a TCP/RED fluid-flow model is proposed. Ordinary differential equations directly derived from fluid-flow models are numerically solved for performing flow-level simulation. However, the numerical computation algorithm for ordinary differential equations in [39] is quite simple; i.e., network states of fluid-level simulation are updated every fixed step time. Network states are updated even when network states are unchanged, causing slowdown of fluid-level simulation. Also, the TCP fluid-model developed in [39] does

not model the TCP timeout mechanism, so that accuracy of the fluid-flow model is not satisfactory for simulating a rather congested network.

In [40], another TCP/RED fluid-flow model is derived, which models the TCP timeout mechanism. By comparing simulation results with analytic ones, the authors of [40] show it has higher accuracy than the fluid-flow model derived in [39]. Both input and output of fluid-flow models derived are uniformly defined as packet transmission rate. So, those fluid-flow models can be easily interconnected for building the fluid-flow model of a large-scale network. In this chapter, we utilize those fluid-flow models and the modeling approach in [40] for realizing an accurate flow-level simulator for a large-scale network.

In [62], a hybrid system model is proposed. The hybrid system model switches multiple fluid-flow models according to the TCP operation phase (i.e., slow-start phase and congestion avoidance phase) for improving the modeling accuracy. The hybrid system is numerically solved for performing flow-level simulation. In [63], a real-time network simulator is proposed. The real-time network simulator uses a hybrid model which combines time-stepping fluid-flow models with a discrete-event packet-level simulation for improving the modeling accuracy. Ordinary differential equations directly derived from fluid-flow models in the real-time network simulator are numerically solved using the numerical computation algorithm. However, similar to [39], the numerical computation algorithm in [62, 63] is quite simple; network states of fluid-level simulation are updated every fixed step time. Network states are updated even when network states are unchanged, causing slowdown of fluid-level simulation.

In [12, 64], hybrid simulators combining fluid and packet-level simulators have been developed. By partially performing packet-level simulation,

a hybrid simulator is able to measure packet-level performance metrics. Researches of a hybrid simulator does not aim to accelerate flow-level simulation. To realize simulation of a large-scale network, flow-level simulation should be accelerated.

4.2 FSIM (Flow-level SIMmulator)

The notable feature of our flow-level simulator FSIM is fast simulation execution compared with conventional flow-level simulators [39, 65]. For accelerating simulation execution, our flow-level simulator FSIM adopts an adaptive numerical computation algorithm for ordinary differential equations. Another features of our flow-level simulator FSIM are its accuracy and its compatibility with other network performance analysis tools. For improving simulation accuracy, our flow-level simulator FSIM utilizes accurate fluid-flow models [40]. Also, the flow-level simulator FSIM can input and output files compatible with ns-2 [13], which is one of representative packet-level simulators.

In what follows, details of our flow-level simulator FSIM — fluid-flow models, the adaptive numerical computation algorithm for ordinary differential equations, routing, and compatibility with an existing network performance analysis tool — are explained.

4.2.1 Fluid-flow models

FSIM utilizes the fluid-flow model of the TCP congestion control mechanism derived in [40]. Definition of symbols (i.e., constants and variables) used throughout this chapter are summarized in Tab. 4.1.

In the fluid-flow model of the TCP congestion congestion control mech-

Table 4.1: Definitions of symbols (constants and variables)

$x(t)$	input (packet transmission rate)
$y(t)$	output (packet transmission rate)
$R(t)$	TCP round-trip time
$p_{TO}(t)$	TCP timeout probability
min_{th}	minimum threshold value
max_{th}	maximum threshold value
max_p	maximum packet marking probability
α	weight of exponential moving average
c	processing speed of RED router
$q(t)$	current queue length of RED router
$r(t)$	average queue length of RED router
$p_q(t)$	packet marking probability of RED router
$p(t)$	packet loss probability of RED router
τ	propagation delay of link
M	number of TCP flows in the network
N	number of RED routers in the network

anism, the input $x(t)$ is the arrival rate of ACK packets and the output $y(t)$ is the transmission rate of data packets, which is given by

$$\begin{aligned} \dot{y}(t) &= f_1(t, x(t), y(t), y(t - R(t))) \\ &= \frac{x(t)}{y(t)\,R(t)^2} - \frac{2}{3} y(t)\,z(t)\,(1 - p_{TO}(t)) \\ &\quad - \left(\frac{4}{3}\,y(t) - \frac{1}{R(t)}\right) z(t)\,p_{TO}(t), \end{aligned} \tag{4.1}$$

where $z(t) \equiv y(t - R(t)) - x(t)$. $p_{TO}(t)$ is the probability that a packet loss is detected by the timeout mechanism rather than duplicate ACKs, and it can be approximated as $p_{TO}(t) \simeq min(1, 3/w(t))$. $R(t)$ is the TCP round-trip time, which is given by the sum of propagation delays and queuing delays on the path.

FSIM utilizes the fluid-flow model of the RED router derived in [40]. In the fluid-flow model of the RED router, the input $x(t)$ is the arrival rate of data packets and the output $y(t)$ is the transmission rate of data packets,

which is given by

$$\begin{aligned} y(t) &= g_1(t, x(t)) \\ &= \min(c, (1 - p(t))\, x(t)), \end{aligned} \tag{4.2}$$

where $p(t)$ is the packet loss probability. $p(t)$ is given by the packet marking probability $p_q(t)$ [40, 39], the current queue length $q(t)$, and the average queue length $r(t)$:

$$p(t) = \begin{cases} \frac{2}{3} p_q(t) & \text{if } \textit{Wait option on} \\ \frac{2p_q(t)}{1 + p_q(t)} & \text{otherwise} \end{cases} \tag{4.3}$$

$$p_q(t) = \begin{cases} 0 \text{ if } r(t) < min_{th} \\ \frac{max_p}{max_{th} - min_{th}}(r(t) - min_{th}) \\ \quad \text{if } min_{th} \leq r(t) < max_{th} \\ \frac{1 - max_p}{max_{th}} r(t) - (1 - 2max_p) \\ \quad \text{if } max_{th} \leq r(t) < 2max_{th} \\ 1 \text{ if } r(t) \geq 2max_{th} \end{cases} \tag{4.4}$$

$$\dot{r}(t) = -\alpha\, c(r(t) - q(t)) \tag{4.5}$$

$$\dot{q}(t) = \begin{cases} x(t) - c & \text{if } q(t) > 0 \\ max(x(t) - c, 0) & \text{otherwise} \end{cases} \tag{4.6}$$

The link propagation delay is modeled as

$$\begin{aligned} y(t) &= h_1(t, x(t)) \\ &= x(t-\tau), \end{aligned} \tag{4.7}$$

where the input $x(t)$ is the packet transmission rate, output $y(t)$ is the packet transmission rate and τ is the propagation delay for the link.

An entire network is modeled with the analysis technique proposed in [40] by connecting models of the TCP congestion control mechanism, the RED router and the link propagation delay. When the RED router has multiple input links, it is modeled as link convergence from individual input links. Link convergence can be described as the sum of packet transmission rates from individual links. In other words, when the transmission rate for each link is $x_i(t)(1 \leq i \leq L_I)$ where L_I is the number of input links, the transmission rate $y(t)$, is given by

$$y(t) = \sum_{i=1}^{L_I} x_i(t) \tag{4.8}$$

. When the RED router has multiple output links, output from the RED router is modeled through distribution in multiple flows. Flow distribution can be described by distribution of incoming traffic to L_O links where L_O is the number of output links. Let the departure rate of data packets from the RED router be $x(t)$, the outgoing link rate $y_i(t)(1 \leq i \leq L_O)$, and the flow distribution ratio $f_i(t)(1 \leq i \leq L_O)$ for the output link, we have the following equation.

$$y_i(t) = f_i(t)\, x(t) \tag{4.9}$$

4.2.2 Adaptive numerical computation algorithm

In fluid-flow models explained in Section 4.2.1, network state is represented by TCP packet transmission rates $y(t)$, current queue lengths $q(t)$ of RED routers, and the average queue lengths $r(t)$ of RED routers. Let $\mathbf{z}(t)$ be the state vector of a network given by

$$\mathbf{z}(t) = \begin{pmatrix} y_1(t) \\ \vdots \\ y_M(t) \\ y_1(t - R_1(t)) \\ \vdots \\ y_M(t - R_M(t)) \\ r_1(t) \\ \vdots \\ r_N(t) \\ q_1(t) \\ \vdots \\ q_N(t) \end{pmatrix}, \tag{4.10}$$

where M is the number of TCP flows in the network, and N is the number of RED routers in the network. The state vector at $t + \Delta$ is approximately given by

$$\mathbf{z}(t + \Delta) \simeq \mathbf{z}(t) + \dot{\mathbf{f}}(t, \mathbf{z}(t))\, \Delta, \tag{4.11}$$

where $\dot{\mathbf{f}}$ is obtained from fluid-flow models (Eqs. (4.1)–(4.6)).

Using a numerical computation algorithm for ordinary differential equa-

tions, evolution of the network state starting from an initial state can be numerically obtained. As a numerical solution for Eq. (4.11), FSIM utilizes a numerical computation algorithm for ordinary differential equations called Dormand-Prince method [66]. For accelerating simulation execution, FSIM uses the adaptive stepsize control for the Dormand-Prince method [66], which adjusts the stepsize according change in ordinary differential equations. In other words, when change in the network state is large, the stepsize is decreased for minimizing error in the numerical computation. On the contrary, when change in the network state is small, the stepsize is increased for speeding up the numerical computation. With such an adaptive control, computational complexity required for flow-level simulation can be significantly reduced, while maintaining the accuracy of simulation results.

Notice that the fluid-flow model of the TCP congestion control mechanism (Eq. (4.1)) requires past network state (i.e., $y(t - R)$). In FSIM, past network states (up to the maximum round-trip time of all TCP flows) are recorded in the memory for enabling application of the Dormand-Prince method. Since FSIM uses the adaptive stepsize control, the timing at which the network state is updated is varied. So past network state required for calculating the next network state might not have been calculated. In FSIM, the past network state in need is approximated as an interpolation of nearby network states [67].

4.2.3 Routing

FSIM obtains routing information using a independent module. To obtain routing information, the module performs a single-source shortest path

algorithm. FSIM would be able to support dynamic routing by dynamically modifying routing information depending on a dynamic routing algorithm.

4.2.4 Compatibility with existing performance evaluation tools

The flow-level simulator FSIM realizes high compatibility with an existing network performance evaluation tool. Specifically, FSIM can input and output files compatible with ns-2 [13], which is one of the most popular packet-level simulators. Specifically, our flow-level simulator FSIM interprets typical ns-2 simulation file written in OTcl[68]; i.e., major ns-2 commands such as `duplex-link` and `create-connection` are parsed and translated to FSIM objects. Also, FSIM outputs its simulation logs in either ns-2 (generated with `trace-all` command) or nam [69] (generated with `namtrace-all` compatible format.

4.3 Experiments

In this section, through extensive experiments using our FSIM implementation, we evaluate the effectiveness of our flow-level simulator FSIM in terms of accuracy, simulation speed, and memory consumption.

4.3.1 Experimental setup

We compare performance of three simulators: our flow-level simulator FSIM, the conventional flow-level simulator FFM [70], and packet-level simulator ns-2 [13]. FFM is one of the latest flow-level simulators. Recall the major difference between FFM and FSIM; i.e., different from FSIM, FFM has no adaptive stepsize control, and the TCP fluid-flow model in

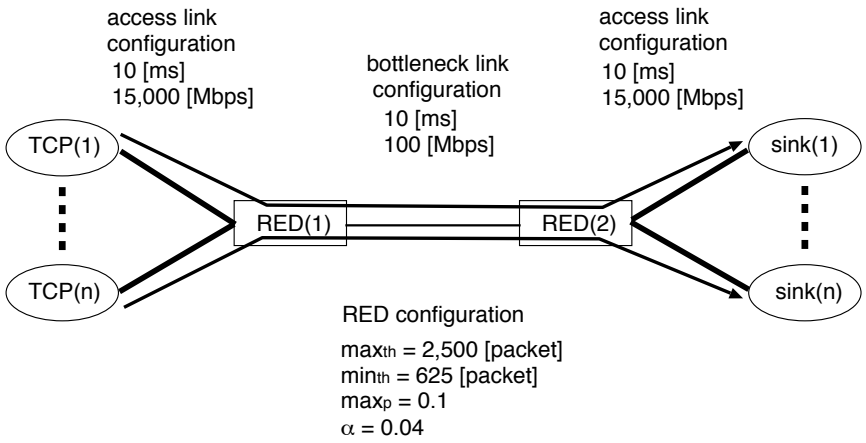


Figure 4.1: Dumb-bell network

FFM does not model the TCP timeout mechanism. We performed simulations for the same topology and parameters with those three simulators.

It is essential to investigate the performance of a network simulator for several simulation scenarios since network simulators are in nature used for several purposes. We therefore use three typical simulation scenarios: dumb-bell network, random network, and hierarchical network.

- Dumb-bell network

  A dumb-bell network [71] consists of two RED routers and homogeneous TCP flows with identical propagation delays (see Fig. 4.1). The link between two RED routers is the bottleneck since the access link (i.e., the link between an end host and an RED router) is much faster than that between RED routers. Unless explicitly stated, the number of TCP flows is 100, and the bottleneck link bandwidth is 100 [Mbit/s]. Such a simple simulation scenario has been widely adopted as a baseline model for many networking performance studies [71].

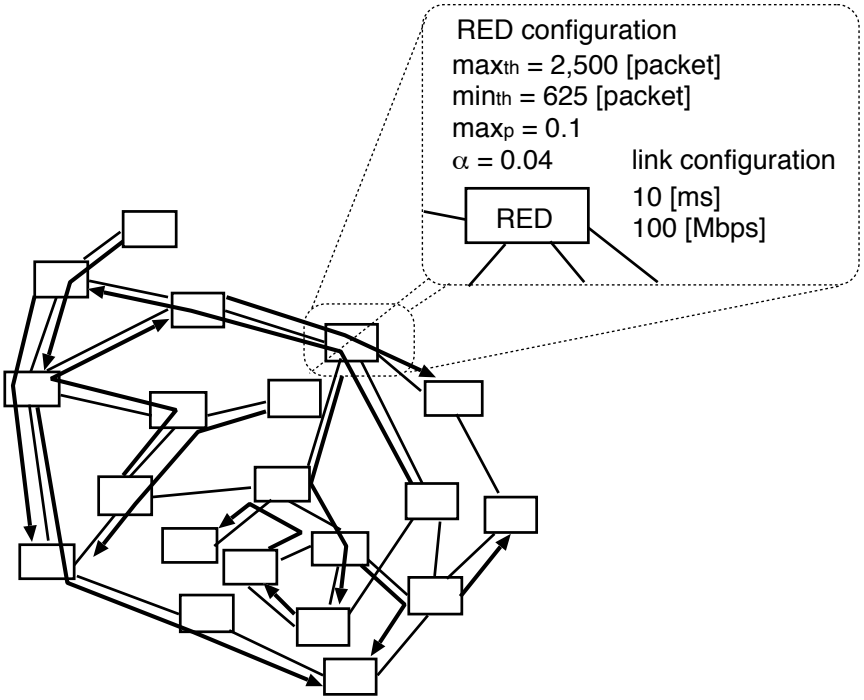


Figure 4.2: A random network with 20 nodes

- Random network

  In the random network scenario (see Fig. 4.2), for a given number of RED routers and the average degree (i.e., the average number of links connected to an RED router), a network topology is randomly generated as a random network [72]. Also, for a given number of TCP flows, TCP source hosts and sinks are attached at randomly-chosen RED routers. The random network is widely used by large-scale network researchers [73-75]. Unless explicitly stated, the bandwidth of all links are equally set to 100 [Mbit/s], the propagation delay of all links are equally set to 10 [ms], and the average degree is 3.

- Hierarchical network

  The hierarchical network consists of three levels, are referred to as

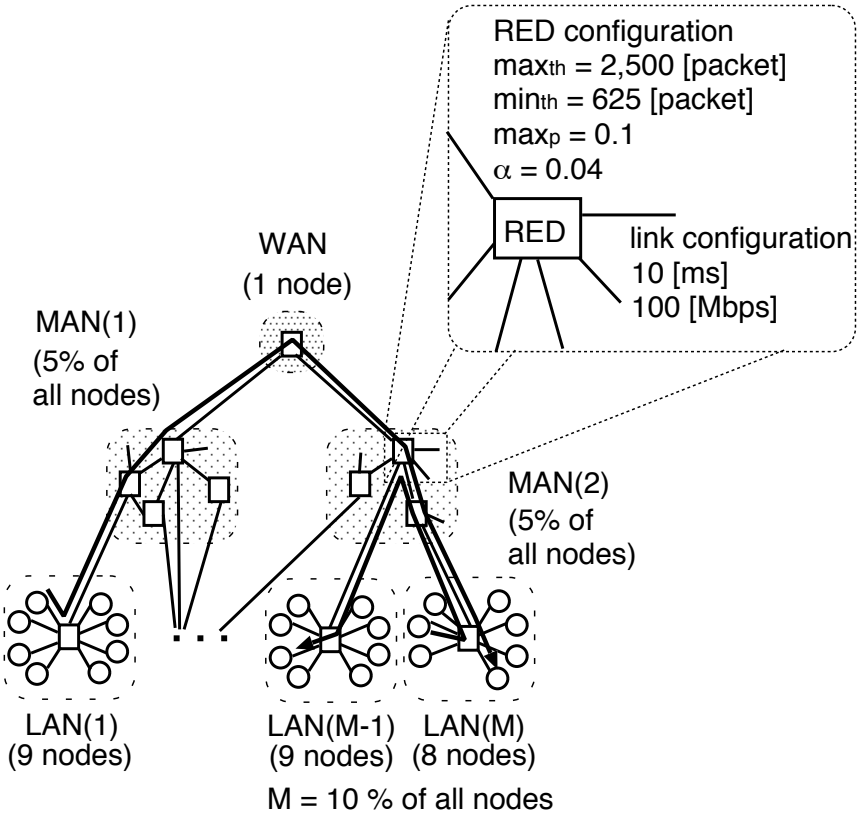


Figure 4.3: Hierarchical network

WAN, MAN and LAN levels (see Fig. 4.3). The hierarchical network is a Internet-like network [76]. For a given number of RED routers, a network topology is randomly generated as a hierarchical network [76]. Then, for a given number of TCP flows, TCP source hosts and sinks are attached at randomly-chosen RED routers at LAN level. Unless explicitly stated, the bandwidth of all links are equally set to 100 [Mbit/s], and the propagation delay of all links are equally set to 10 [ms],

In all simulation scenarios, all TCP source hosts continuously transmit data to their corresponding TCP sinks.

In all experiments, a computer with two Intel Pentium 4 (3.06 [GHz]) processors with 3 [GByte] memory running Debian GNU/Linux 3.1 (kernel version 2.4.32) is used for executing flow-level simulators or the packet-level simulator.

In all experiments, we repeated 10 simulations, and calculated the average and 95% confidence interval of measurements (e.g., simulation execution time and maximum memory consumption). In the following figures, confidence intervals are not shown because they were sufficiently small in all experiments. Note that we optimized ns-2 configurations following the guideline in [77].

4.3.2 Accuracy

With three network simulators (i.e., FSIM, FFM, and ns-2), TCP packet transmission rate and the queue length of the RED router are measured (Figs. 4.4 through 4.6). For different bottleneck link bandwidths, Figs. 4.4 and 4.5 illustrate the time average of TCP packet transmission rates and the relative error of TCP packet transmission rates between a flow-level simulator (FSIM or FFM) and the packet-level simulator ns-2, respectively. Figure 4.5 shows that FSIM achieves much better accuracy than FFM. Note that Fig. 4.5 shows that the accuracy of FSIM improves as the bottleneck link bandwidth increase. Therefore, FSIM is effective for high-speed network simulations which is hard to perform with packet-level simulators.

4.3.3 Simulation speed

We investigate simulation speeds of flow-level simulators (FSIM and FFM) and the packet-level simulator ns-2 when changing the number of TCP

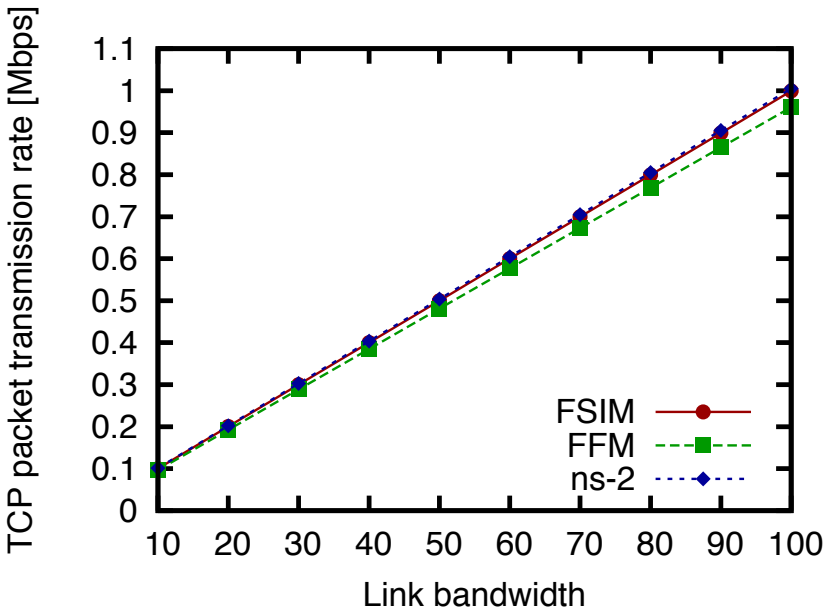


Figure 4.4: Time average of TCP packet transmission rates vs. the link bandwidth

flows, the link bandwidth, and the number of nodes. To investigate the simulation speed, we measured simulation execution times of three simulators. With three simulators, simulation execution times required for performing 50 [s] of simulation are measured. The purpose of flow-level simulator is different with that of packet-level simulator. For reference, we present results of the packet-level simulator ns-2.

We first investigate simulation speeds of three simulators when changing the number of TCP flows in the dumb-bell network. Figure 4.7 shows simulation execution times of three simulators for different numbers of TCP flows.

Figure 4.7 shows that the simulation execution time of FSIM is much shorter than that of FFM regardless of the number of TCP flows. This is because the adaptive stepsize control implemented in FSIM is effective re-

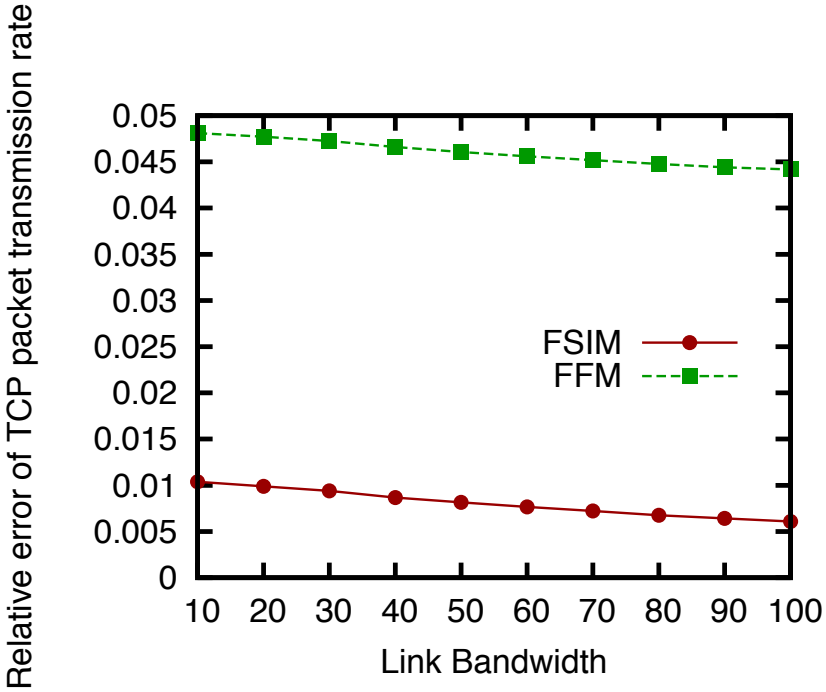


Figure 4.5: Relative error of the time average of TCP packet transmission rates vs. the link bandwidth

gardless of the number of TCP flows. Figure 4.7 also shows that simulation execution times of FSIM and FFM does not increase even when the number of TCP flows increases. This is because flow-level simulators support *flow aggregation*, which aggregates multiple flows with the same characteristics into a single one. Figure 4.7 even shows that the simulation execution time of FSIM decreases as the number of TCP flows increases. This can be explained as follows. As the number of TCP flows increases, the TCP round-trip time increases, leading gradual change in the TCP transmission rate. As the change in the TCP window size becomes gradual, the stepsize increase, which results in fast simulation.

We then investigate simulation speeds of three simulators when changing the bottleneck link bandwidth in the dumb-bell network. Figure 4.8 shows simulation execution times of three simulators for different bottle-

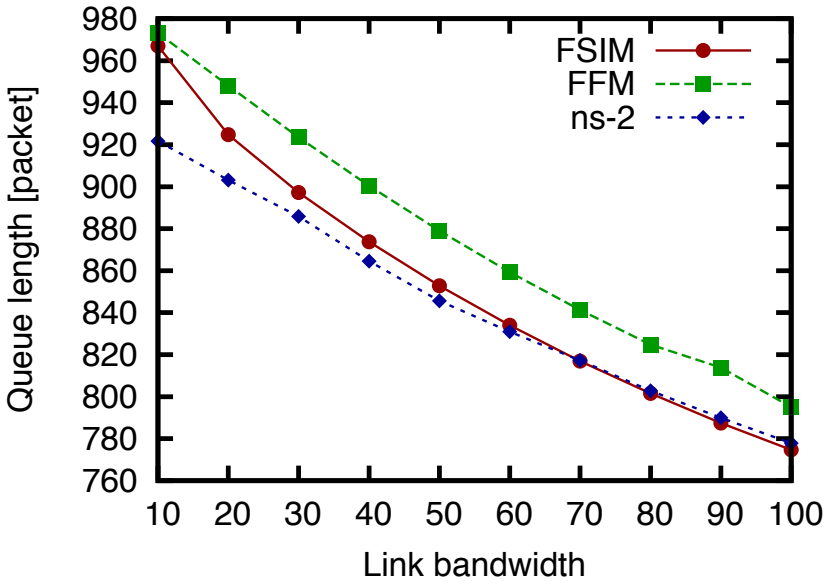


Figure 4.6: Time average of the queue length vs. the link bandwidth

neck link bandwidths.

Figure 4.8 shows that the simulation execution time of FSIM is much shorter than that of FFM and ns-2 regardless of the bottleneck link bandwidth. Surprisingly, Fig. 4.8 shows that the simulation execution time of FSIM even decreases as the bottleneck link bandwidth increases. This is resulted from our adaptive stepsize control. Namely, as the bottleneck link bandwidth increase, the linear growth time of the TCP transmission rate becomes long. Thus, the stepsize can be increased when the bottleneck link bandwidth is large. Figure 4.8 shows that simulation execution times of FSIM are almost the same if the bottleneck link bandwidth is more than or equal to 1,000 [Mbit/s]. This can be explained as follows. The ordinary differential equations (Eqs. (4.1) and (4.7)) used in FSIM are dependent on the past network state at the delay time ago. Therefore, when the gradient of network state from an ordinary differential equation is calculated, the

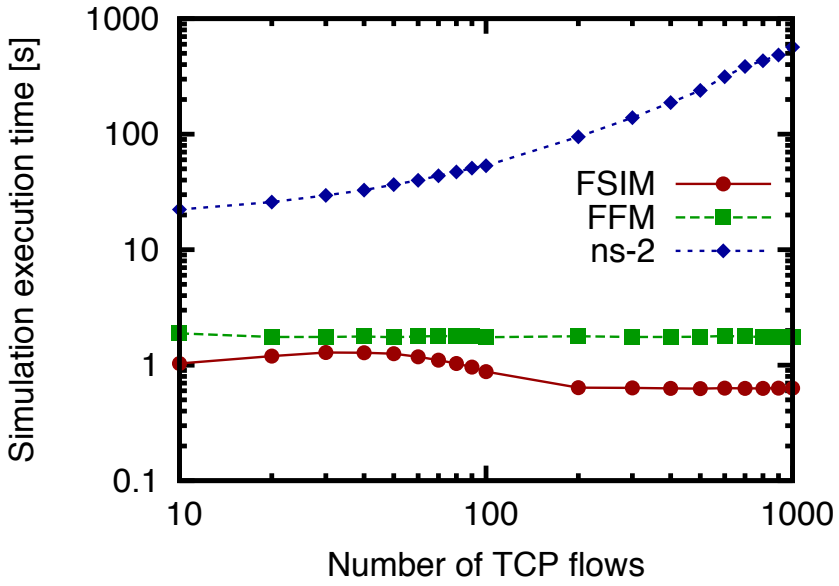


Figure 4.7: Simulation execution time vs. the number of TCP flows

past network state at the delay time ago must be already calculated. The stepsize in FSIM cannot be larger than the delay time. The reason that simulation execution times of FSIM are almost the same is that the stepsize become the delay time. Figure 4.8 shows that simulation execution times of FSIM and FFM does not increase even when the bottleneck link bandwidth increases. This clearly indicates the strength of flow-level simulators; the computational complexity of flow-level simulators does not increase as the link bandwidth increases.

We then investigate simulation speeds of three simulators when changing the number of nodes. Figures 4.9 and 4.10 show simulation execution times of three simulators for different numbers of nodes in the random network and the hierarchical network, respectively. Due to memory exhaustion, we could not perform simulation 900 nodes with FFM and ns-2. We will discuss this issue in Section 4.3.4.

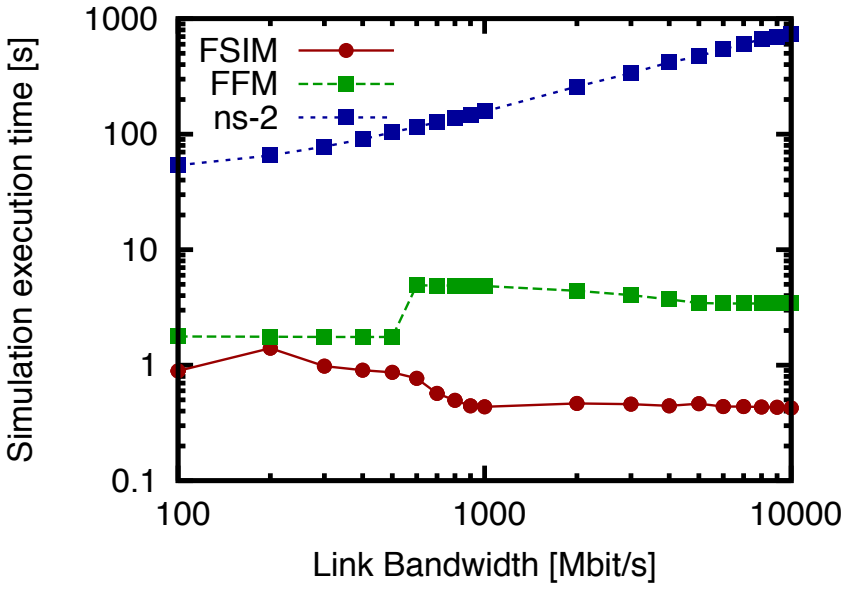


Figure 4.8: Simulation execution time vs. the bottleneck link bandwidth

Figures 4.9 and 4.10 show that the simulation execution time of FSIM is much shorter than that of FFM and ns-2 regardless of the number of nodes. From Figs. 4.9 and 4.10, it is found that the difference between simulation execution times of FFM and FSIM for a random network is smaller than that for a hierarchical network. This is because that it is easy to synchronize the network state in a hierarchical network compared with a random network. Since traffic concentration at top node is the highest in a hierarchical network, it is easy to find a bottleneck link of a flow. Therefore, the network state in a hierarchical network tends to be easily synchronized compared with a random network. Figures 4.9 and 4.10 also show that simulation execution times of FSIM and FFM increase when the number of nodes increases. This is because the number of TCP flows with a different path increases when the number of nodes increases. TCP flows with a different path cannot be aggregated into a single one by flow-aggregation. Thus, the

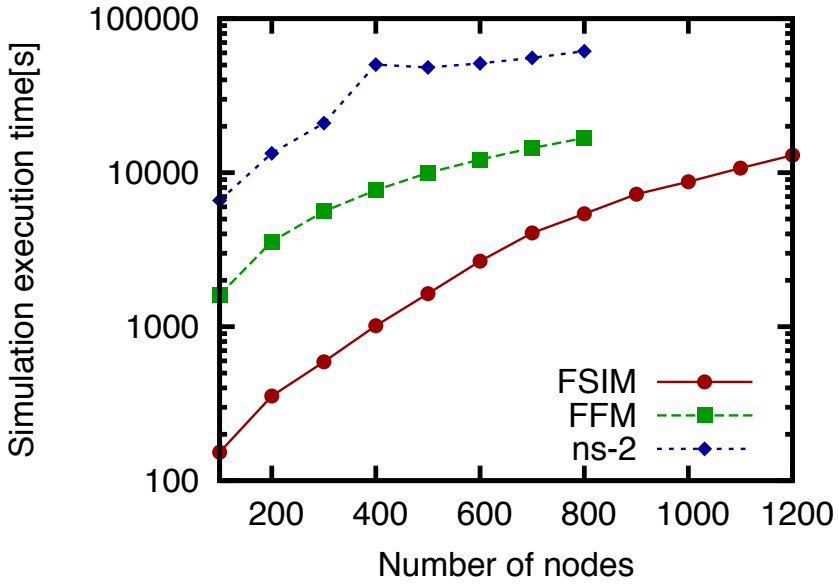


Figure 4.9: Simulation execution time vs. the number of nodes in the random network

computational complexity of FSIM and FFM increases when the number of nodes increases. Figure 4.10 shows that simulation execution times of flow-level simulators and the packet-level simulator increases almost linearly. Hence, relative execution times between flow-level simulators and packet-level simulator for different number of nodes are almost the same. Therefore, the effectiveness of flow-level simulator does not become high as the number of nodes increases.

In summary, the simulation speed of FSIM is much faster than that of FFM regardless of the number of TCP flows, the link bandwidth, and the number of nodes. FSIM realizes 2-3 times faster simulation compared with FFM on the average.

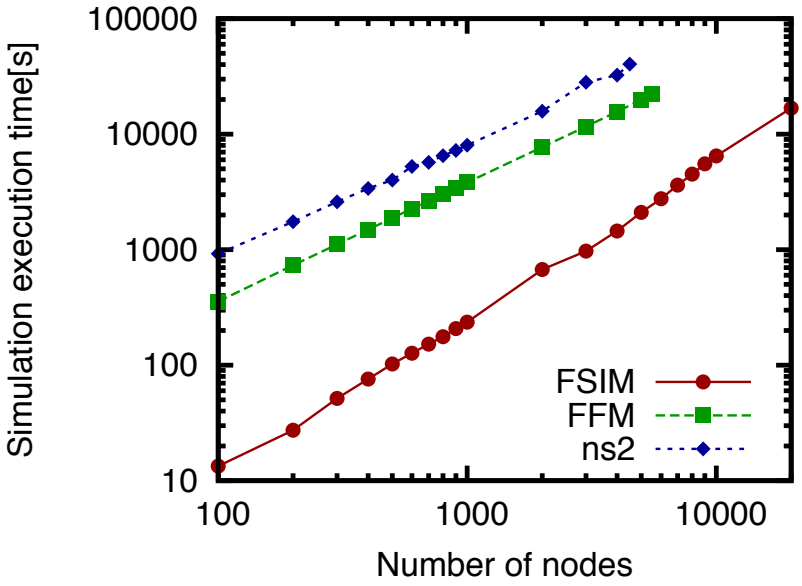


Figure 4.10: Simulation execution time vs. the number of nodes in the hierarchical network

4.3.4 Memory consumption

We investigate memory consumptions of flow-level simulators (FSIM and FFM) and the packet-level simulator ns-2 when changing the number of TCP flows, the link bandwidth, and the number of nodes. Scalability of network simulators is sometimes limited by the memory size required for executing simulation [77]. To investigate the memory consumption, maximum memory consumptions (i.e., the sum of statistically and dynamically allocated memory size) during simulation run are measured for three simulators.

We first investigate memory consumptions of three simulators when changing the number of TCP flows in the dumb-bell network.

Figure 4.11 shows that the maximum memory consumptions of FSIM is much smaller than that of FFM regardless of the number of TCP flows.

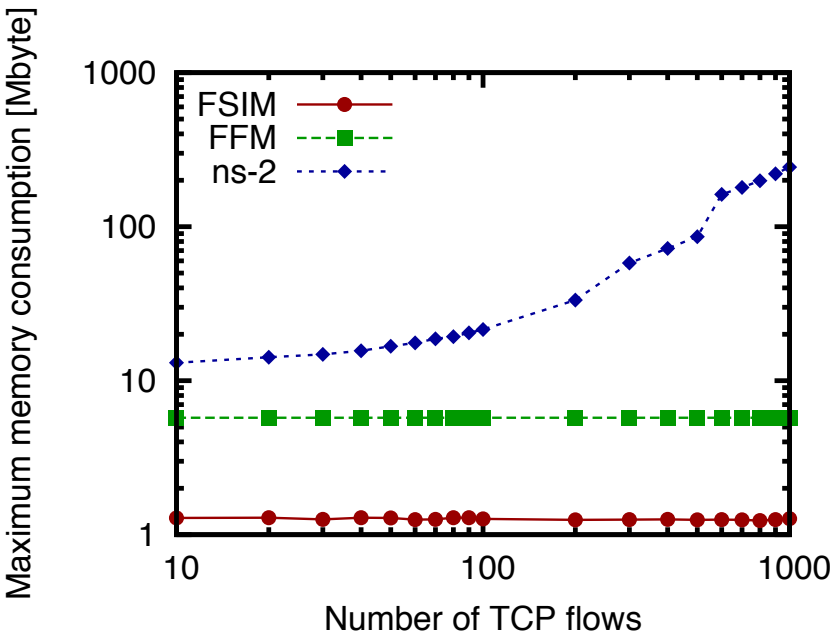


Figure 4.11: Maximum memory consumption vs. the number of TCP flows

This can be explained as follows. FFM is embedded in ns-2, which has many functions not to be related to flow-level simulation. FFM loads such functions on a memory during execution. It is not trivial to modify FFM to load functions required only for flow-level simulation due to its high module coupling. Figure 4.11 also shows that maximum memory consumptions of FSIM and FFM does not increase even when the number of TCP flows increases. Similar to the phenomenon observed in Fig. 4.7, this is because flow-level simulators support flow-aggregation.

We then investigate memory consumptions of three simulators when changing the bottleneck link bandwidth in the dumb-bell network.

Figure 4.12 shows that the maximum memory consumption of FSIM is independent of much smaller than that of FFM and ns-2 regardless of the bottleneck link bandwidth. Similar to the phenomenon observed in Fig. 4.8, this is because flow-level simulators mimics behavior of every flow in a

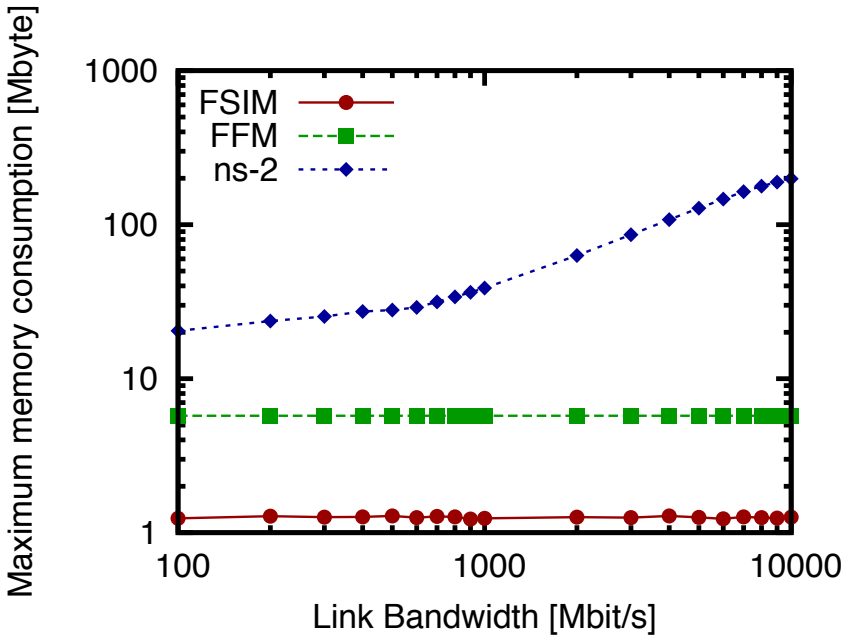


Figure 4.12: Maximum memory consumption vs. the bottleneck link bandwidth

network.

We then investigate memory consumptions of three simulators when changing the number of nodes. Figures 4.13 and 4.14 show maximum memory consumptions of three simulators for different numbers of nodes in the random network and the hierarchical network, respectively.

Figures 4.13 and 4.14 show that the maximum memory consumption of FSIM is much smaller than that of FFM regardless of the number of nodes. This is because FSIM stores the the path of a TCP flow as a list with the number of elements equal to the path length whereas FFM stores the path of a TCP flow in a array with the number of elements equal to the total number of nodes. The maximum memory consumption of FSIM when performing simulation with 10,000 nodes is approximately 1,430 [Mbyte]. Figures 4.13 and 4.14 also show that maximum memory consumptions of

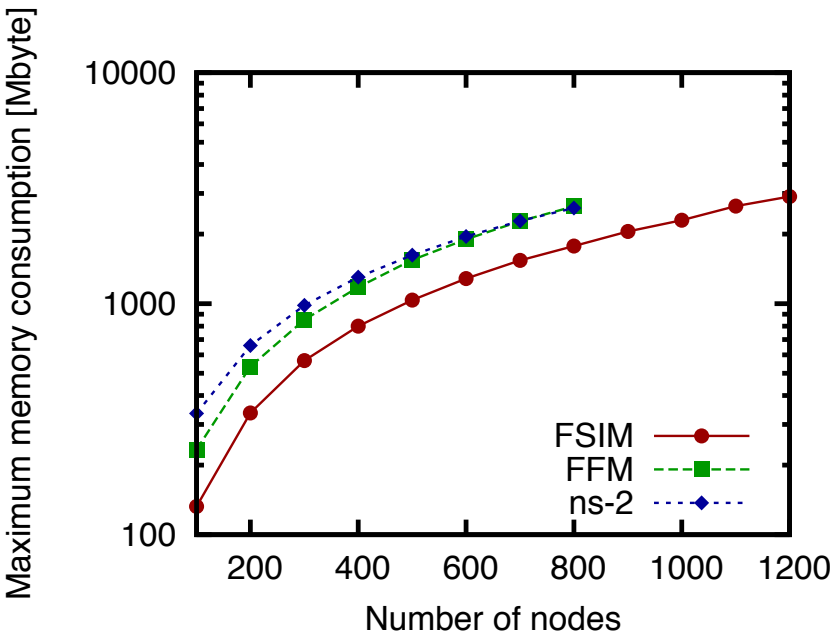


Figure 4.13: Maximum memory consumption vs. the number of nodes in a random network

FSIM and FFM increase as the number of nodes increases. Similar to the phenomenon observed in Figs. 4.9 and 4.10 this is because the number of TCP flows with a different path increases when the number of nodes increases.

In summary, the memory consumption of FSIM is always smaller than that of FFM regardless of the number of TCP flows, the link bandwidth, and the number of nodes. FSIM realizes more than 2-4 times memory efficiency compared with FFM. This suggests that for a given memory size, our flow-level simulator FSIM can simulate a larger network than FFM.

4.4 Summary

In this chapter, we have proposed a flow-level simulator called FSIM (Fluid-based SIMulator) for performance evaluation of large-scale networks, and

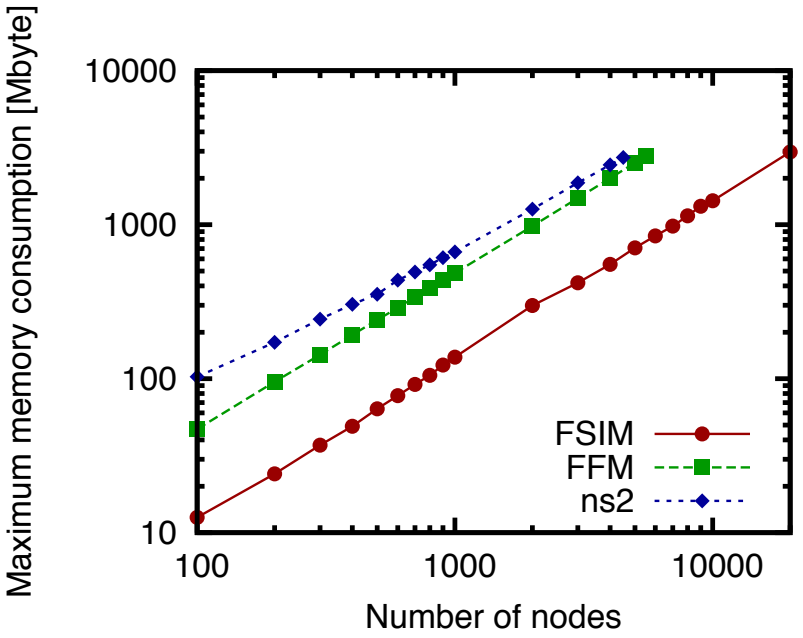


Figure 4.14: Maximum memory consumption vs. the number of nodes in a hierarchical network

have verified its effectiveness using our FSIM implementation. Through extensive experiments using our FSIM implementation, we have evaluated the effectiveness of our flow-level simulator FSIM in terms of simulation speed, accuracy and memory consumption. We have shown that our flow-level simulator FSIM outperforms a conventional flow-level simulator; i.e., it realizes approximately 200%-300% faster simulation with higher accuracy and less memory consumption than a conventional flow-level simulator. In particular, we should note that FSIM is effective for performance evaluation of a network with large link capacities and many TCP flows.

As future work, we are planning to further improve the numerical computation algorithm of differential equation. In particular, we are planning to further discuss the adaptive numerical computation algorithm used in FSIM. We are also planning to evaluate FSIM in a large-scale network in

terms of accuracy. We are planning to include support for various types of network protocols such as UDP, DCCP, HighSpeed TCP, and XCP utilizing fluid-flow models derived in [78, 79, 33].

Our FSIM implementation is available at `http://www.ispl.jp/fsim/`.

Chapter 5

Effectiveness of Thorup's Shortest Path Algorithm for Large-Scale Network Simulation

5.1 Thorup's Algorithm

In this section, we briefly introduce Thorup's algorithm. Refer to [48] for the details.

Thorup's algorithm is an efficient solution of a single-source shortest path (SSSP) problem for an undirected graph with positive integer edge weights [48]. In Dijkstra's algorithm, vertices are visited in order of increasing distance from the source vertex. Thus, Dijkstra's algorithm need to sort vertices according to their distances from the source vertex, which results in non-linear time complexity. Thorup's algorithm can realize a linear time

complexity by avoiding the sorting bottleneck. Several key techniques used in Thorup's algorithm are a hierarchical bucketing structure and identification of vertex pairs that can be visited in any order. Note that edge weights are restricted to positive integers due to introduction of buckets.

Thorup's algorithm is composed of two phases: building the component tree and visiting components for finding shortest-paths.

First, Thorup's algorithm builds a hierarchical bucketing structure called *component tree*, which is suitable for finding shortest paths from a source vertex in the next phase. The graph is recursively partitioned into several subgraphs. Namely, the graph is partitioned into several subgraphs by removing edges having a large weight. This procedure is repeated (i.e., subgraphs are further partitioned into small subgraphs) until all subgraphs become singletons. A component (i.e., node) of the component tree represents each subgraph; i.e., the root component of the component tree represents the whole graph, and leaf components of the component tree represent vertices in the graph. For efficient realization of recursive partitioning of the graph, Thorup's algorithm utilizes the minimum spanning tree of the graph. Thus, the time complexity of building the component tree is determined by that of the algorithm for obtaining the minimum spanning tree.

Second, Thorup's algorithm finds shortest paths from a source vertex using the component tree. Similarly to Dijkstra's algorithm, Thorup's algorithm obtains the shortest paths from the source vertex by gradually expanding the set of visited vertices. For this purpose, components of the component tree are recursively visited from the root component while identifying vertex pairs that can be visited in any order. A bucket is assigned to each component in the component tree, and its key and values

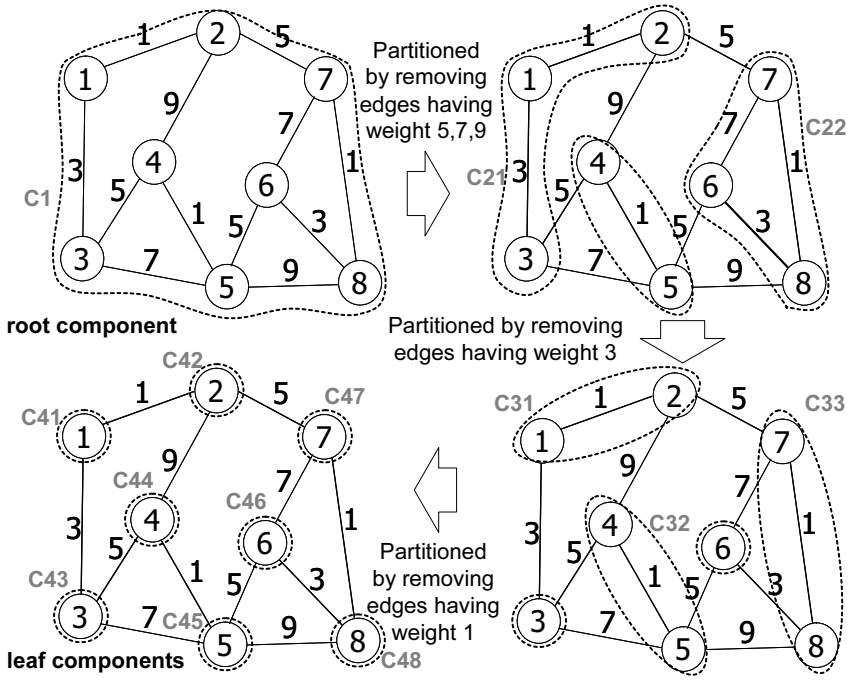


Figure 5.1: Recursive partitions for an example graph

are successively updated. The time complexity of visiting components for finding shortest-paths is $O(E + N)$.

In [48], two algorithms for finding the minimum spanning tree, Kruskal's [50] and Fredman's [49] algorithms, are discussed.

Fredman's algorithm is an efficient algorithm, whose time complexity is $O(E)$. The original Thorup's algorithm (THORUP-FR) uses Fredman's algorithm for obtaining the minimum spanning tree. Incorporation of Fredman's algorithm into THORUP-FR is another key technique for realizing a linear time algorithm of $O(E + N)$.

Kruskal's algorithm is an elementary algorithm with the time complexity of $O(E\,\alpha(N))$ where $\alpha(N)$ is the functional inverse of the Ackerman function [47]. A simplified version of Thorup's algorithm (THORUP-KL)

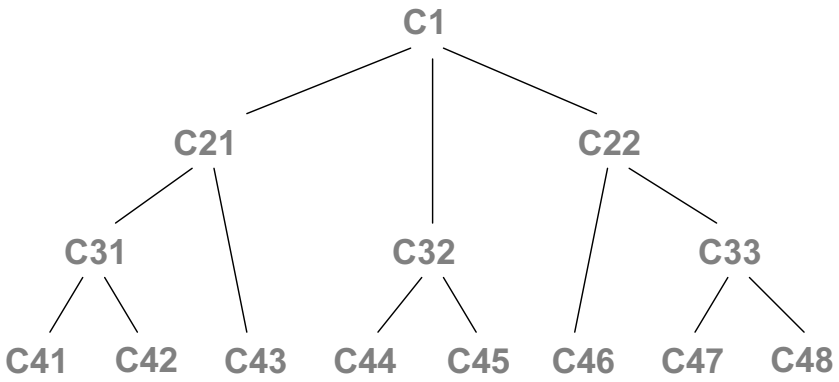


Figure 5.2: Component tree for an example graph

uses Kruskal's algorithm for obtaining the minimum spanning tree. THORUP-KL is no longer a linear time algorithm since Kruskal's algorithm is not a liner time algorithm; i.e., the time complexity of THORUP-KL is $O(E\,\alpha(N)+N)$.

5.2 Experiment

5.2.1 Methodology

Through extensive experiments, we compare the performance of Thorup's and Dijkstra's algorithms in terms of speed and memory consumption. We implemented Thorup's algorithm using Kruskal's algorithm for obtaining the minimum spanning tree (THORUP-KL) and Dijkstra's algorithm using a binary heap (DIJKSTRA-BH) [47].

Note that in our experiments, THORUP-KL is used instead of THORUP-FR even though THORUP-KL is theoretically less efficient than THORUP-FR. As explained in Section 5.1, THORUP-KL is not a linear time algorithm. However, we intentionally use THORUP-KL rather than THORUP-FR be-

cause of the following two practical reasons.

The first reason is that Asano *et al.* have shown that THORUP-FR is very slow in practice [51]. The second reason is that the time complexity of Kruskal's algorithm, $O(E\,\alpha(N))$, should be comparable with that of Fredman's, $O(E)$, since it is known that $\alpha(N) \leq 4$ for all $N < 2^{2^{2^{65536}}} - 3$ [47]. Namely, the performance of THORUP-KL should be comparable with (or might be even faster than) that of THORUP-FR for large-scale network simulation.

In all experiments, a computer with a single Intel Pentium4 2.80 [GHz] processor with 1 [Gbyte] of memory running Debian GNU/Linux 5.0 (kernel version 2.6.26) is used.

For clearly examining effectiveness of THORUP-KL and DIJKSTRA-BH, we used a simple network topology. Namely, we generated a random network with ER (Erdös-Rényl) model [72] for a given network size N (i.e., the number of nodes) and the average degree k (i.e., the average number of links connected to each node). A random network is a traditional network model, and it has been used for large-scale network simulation [80]. We claim neither that a random network is the *typical* network topology for network simulation studies nor that performance evaluation with a random network is sufficient for comparing effectiveness of THORUP-KL and DIJKSTRA-BH. However, we believe that usage of a realistic (but complex) network model as a network topology unnecessary complicates our performance evaluation, and makes it difficult to interpret the results.

We calculated the average and 95% confidence interval of measurements (e.g., execution time and memory consumption). In all figures, 95% confidence intervals are not shown since they are negligibly small (i.e., less than 1.0% of each measurement).

5.2.2 Speed

We first evaluate the effectiveness of THORUP-KL and DIJKSTRA-BH in terms of speed by measuring and comparing their execution times. THORUP-KL and DIJKSTRA-BH are for a single-source shortest path problem. On the contrary, network simulators usually need to obtain shortest paths for all source–destination node pairs. We therefore estimated the execution time for obtaining *all-pairs* shortest paths from the execution time for obtaining single-source shortest paths. Note that obtaining all-pairs shortest paths is the worst case; i.e., if the number of source–destination pairs is not so large, the network simulator does not need to obtain all-pairs shortest paths. We separately measured the mean times for program initialization, $\overline{T^G_{init}}$, and computation of single-source shortest paths, $\overline{T^G_{SSSP}}$, for a graph G. The execution time for obtaining all-pairs shortest paths for G with N vertices, T^G_{APSP}, is estimated as

$$T^G_{APSP} \quad = \quad \overline{T^G_{init}} + N\,\overline{T^G_{SSSP}}.$$

Execution times of THORUP-KL and DIJKSTRA-BH for obtaining all-pairs shortest paths for different network sizes with a fixed average degree (i.e., k = 5) and multiple edge weights (i.e., randomly-chosen integer edge weights from 1 to 1,000) are shown in Fig. 5.3. This figure shows that execution times increase rapidly as the network size N increases. This phenomenon is not surprising since the time complexity of any algorithm for all-pairs shortest path problem is at least $O(N^2)$.

To visually show the difference in execution times of THORUP-KL and DIJKSTRA-BH , the relative execution time (i.e., the execution time of THORUP-KL normalized by that of DIJKSTRA-BH) for $k = 5$ and 10 is shown in

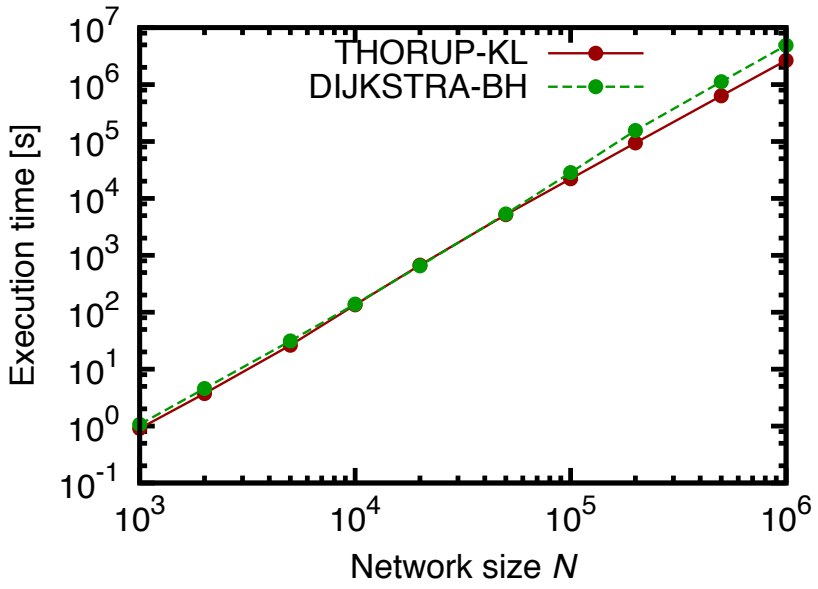


Figure 5.3: Execution times of THORUP-KL and DIJKSTRA-BH for obtaining all-pairs shortest paths for the average degree $k = 5$ and multiple edge weights of 1–1000.

Fig. 5.4. In this figure, results with two types of edge weights are shown: the single edge weight (i.e., 1 for all edges) and multiple edge weights (i.e., randomly-chosen integer edge weights from 1 to 1,000). Figure 5.5 shows relative execution times for random graph and hierarchical graph shown in Fig. 4.3.

Different from the performance comparison of THORUP-FR and DIJKSTRA-FH in [51], this figure clearly indicates that THORUP-KL is almost always faster than DIJKSTRA-BH for any network size, average degree, and edge weights. In [51], it has been reported that even with its linear time complexity, THORUP-FR is at least 10 times slower than DIJKSTRA-FH for medium-scale random graphs with 50,000 vertices. In Fig. 5.4, the maximum value of the relative execution time is 1.02, which implies that the performance of THORUP-KL is comparable with that of DIJKSTRA-BH even

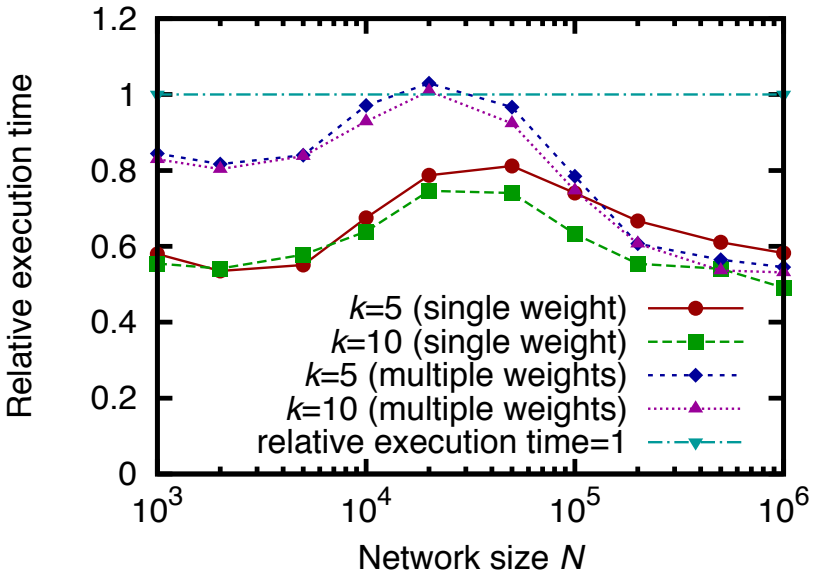


Figure 5.4: Relative execution time (i.e., the execution time of THORUP-KL normalized by that of DIJKSTRA-BH) for different average degrees k and types of edge weights.

in the worst case. For large-scale (e.g., million nodes) network simulation, THORUP-KL achieves almost double efficiency compared with DIJKSTRA-BH.

These results bring us a natural question: why does THORUP-FR perform inefficiently in [51] while THORUP-KL performs efficiently in our experiments? One possible explanation is the algorithm used for finding the minimum spanning tree. Namely, Asano *et al.* used Fredman's algorithm whereas we used Kruskal's algorithm. It has been reported in [51] that for random graphs with 50,000 vertices, Fredman's algorithm itself consumes significant amount of time (e.g., 86% for $E = 175,065$ and 98% for $E = 424,396$) of the total execution time in their experiments.

From detailed measurements of our implementation, we have found that Kruskal's algorithm itself consumed approximately 5% of the total ex-

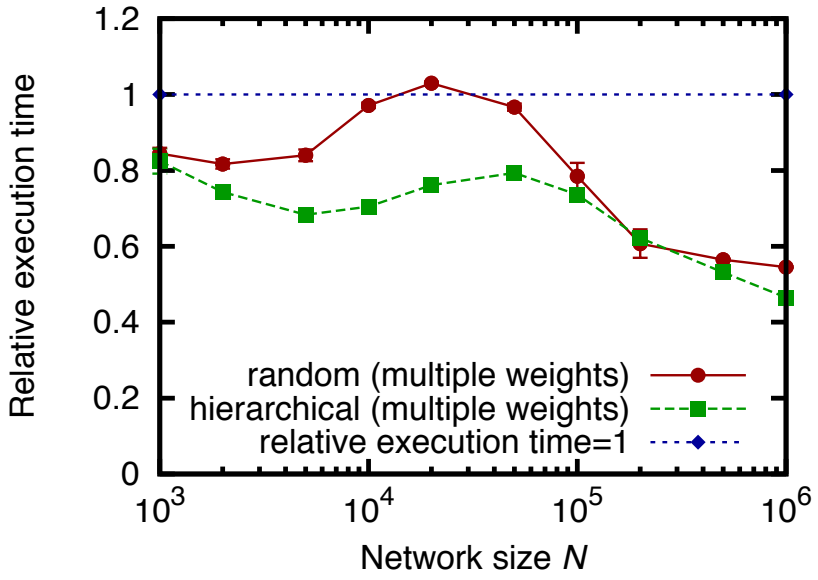


Figure 5.5: Relative execution time for different network topology.

ecution time. Such a drastic reduction in the execution time for finding the minimum spanning tree should be the reason for effectiveness of THORUP-KL. Note that Asano *et al.* have addressed and extensively studied practical inefficiency of Fredman's algorithm in [51]. We could choose THORUP-KL with Kruskal's algorithm mostly because of their findings.

Also, contrary to one's expectation, Fig. 5.4 shows somewhat strange phenomenon; i.e., the relative execution time is like a *camel-shaped function* for the network size N regardless of the average degree k and the type of edge weights.

The relative time complexity of THORUP-KL to DIJKSTRA-BH is given by

$$\frac{O((E\,\alpha(N)+N)}{O((E+N)\,\log N)} \approx O\left(\frac{\alpha(N)}{\log N}\right). \tag{5.1}$$

Note the existence of $\alpha(N)$ in the numerator because we used Kruskal's algorithm instead of Fredman's. Recall that $\alpha(N) \leq 4$ for all $N < 2^{2^{2^{65536}}} - 3$ [47]. Thus, Eq. (5.1) should be an asymptotically monotone decreasing for the network size N. One would therefore expect that the relative execution time T^G_{APSP} be an asymptotically monotone decreasing function for the network size N.

Aside from the camel-shaped function, Fig. 5.4 indicates that the relative execution time is moderately affected by the type of edge weights, and slightly by the average degree. This is again contrary to one's expectations. The relative time complexity (Eq. (5.1)) is independent of the average degree k (i.e., the number of edges E). One would therefore expect that the relative execution time T^G_{APSP} should not be affected by the average degree k.

In the following sections, we will investigate the cause of those deviations of the relative execution time (i.e., camel-shaped function and dependence on the average degree) from the relative time complexity.

5.2.3 Memory Consumption

We next evaluate the effectiveness of THORUP-KL and DIJKSTRA-BH in terms of memory consumption. We measured the memory consumption for obtaining all-pairs shortest paths. Note that the memory consumption for obtaining all-pairs shortest paths is equivalent to that for obtaining single-source shortest paths because of its repetitive program execution.

Memory consumptions of THORUP-KL and DIJKSTRA-BH for obtaining all-pairs shortest paths for different network sizes are shown in Fig. 5.6. In this figure, the average degree $k = 5$ and multiple edge weights of 1–

1000 are used. Note that we have observed that the memory consumptions of THORUP-KL and DIJKSTRA-BH are not significantly affected by the average degree and the type of edge weights.

Figure 5.6 shows that the memory consumption of THORUP-KL is as approximately 1.4 times large as that of DIJKSTRA-BH. This is directly caused by the difference in data structures used in THORUP-KL (i.e., a hierarchical bucketing structure) and DIJKSTRA-BH (i.e., a binary heap). More specifically, the memory complexity of DIJKSTRA-BH is $O(N)$ [46]. The memory complexity of THORUP-KL is also $O(N)$ [48]. Therefore, the relative memory consumption (i.e., the memory consumption of THORUP-KL normalized by that of DIJKSTRA-BH) should be independent of the network size N. Although the data structure of DIJKSTRA-BH (i.e., a binary heap) can be implemented as one-dimensional array, the data structure of THORUP-KL is a complicated hierarchical bucketing structure. A hierarchical bucketing structure consists of a component tree and buckets for each component, which is apparently less memory efficient than the binary heap.

5.2.4 Cause of Camel-Shaped Function

The larger memory consumption with THORUP-KL than with DIJKSTRA-BH might be the cause of the camel-shaped function of the relative execution time (see Fig. 5.4). As observed in Fig. 5.6, the memory consumption of THORUP-KL is as approximately 1.4 times large as that of DIJKSTRA-BH. One possible explanation is the effect of memory cache of the microprocessor. If the memory usage is greedy, memory accesses are less likely to hit the cache, causing significant slow down in program execution.

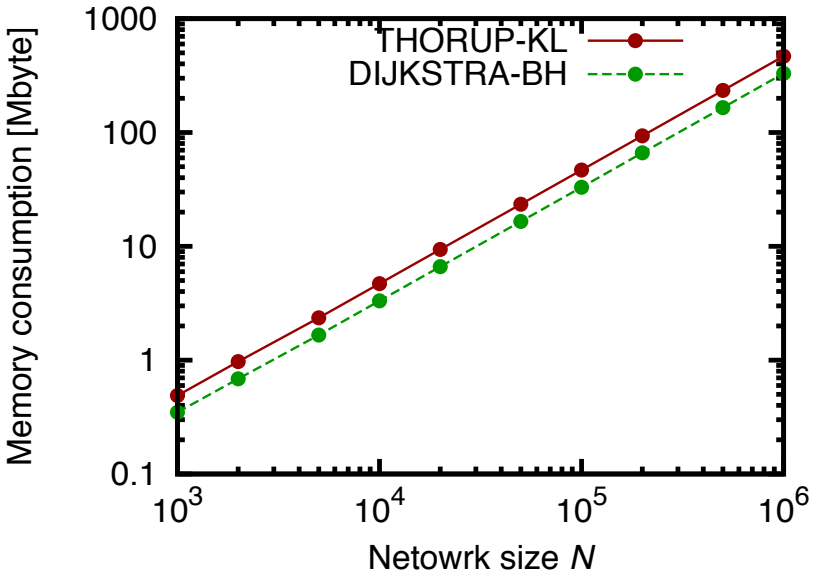


Figure 5.6: Memory consumptions of THORUP-KL and DIJKSTRA-BH for the average degree $k = 5$ and multiple edge weights of 1–1000.

To clarify why the relative execution time becomes a camel-shaped function for the network size N, we investigate the effect of the memory cache on the performance of THORUP-KL and DIJKSTRA-BH. We performed the same experiments with those in Section 5.2.2 except that the memory cache in the microprocessor is disabled. Relative execution times with and without the memory cache are shown in Fig. 5.7.

Figure 5.7 clearly shows that the relative execution time without the memory cache is a monotone decreasing function for the network size N. On the contrary, when the memory cache is enabled, the relative execution time is a camel-shaped function for the network size N.

In Fig. 5.7, the fitted curve with the relative execution time without the memory cache is also plotted. Recall again the relative time complexity (Eq. (5.1)) of THORUP-KL to DIJKSTRA-BH and $\alpha(N) \leq 4$ for all $N <$

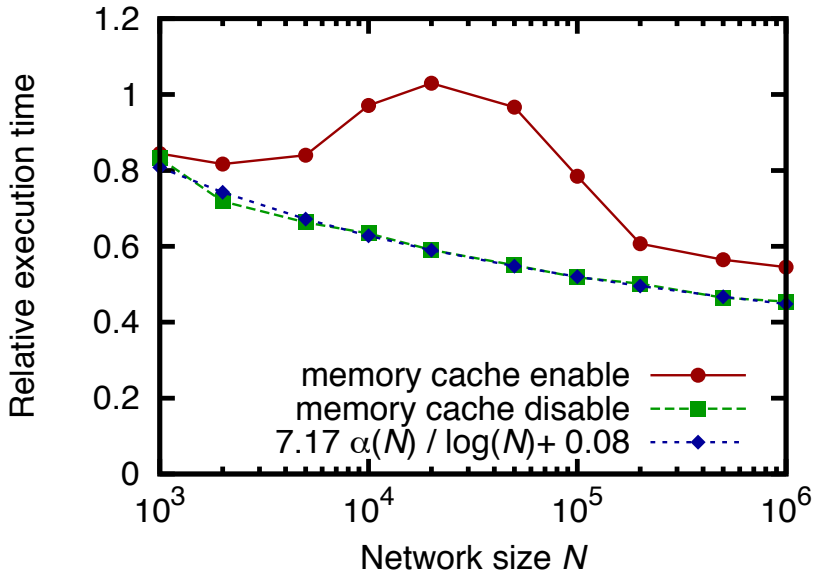


Figure 5.7: Relative execution times with and without the memory cache for the average degree $k = 5$ and multiple edge weights of 1–1000.

$2^{2^{2^{65536}}} - 3$. We therefore used $c_1/\log N + c_2$ for curve fitting, and obtained $c_1 = 7.17$ and $c_2 = 0.08$. The fitted curve well matches the relative execution time without the memory cache, indicating that the relative execution time can be well explained with the relative time complexity (Eq. (5.1)). Reversely, this suggests that the cause of the camel-shaped function of the relative execution time should be the memory cache of the microprocessor.

More detailed investigation can be possible by examining the memory cache performance. We measured the memory cache performance (i.e., L1 and L2 cache miss rates) using a cache profiler called *cachegrind* [81]. Figures 5.8 and 5.9 show cache miss rates of THORUP-KL and DIJKSTRA-BH for L1 and L2 cache, respectively. These figures clearly indicate that THORUP-KL is not cache-friendly. More specifically, L1 and L2 cache miss rates of THORUP-KL is as approximately 2–4 times large as those of DIJKSTRA-

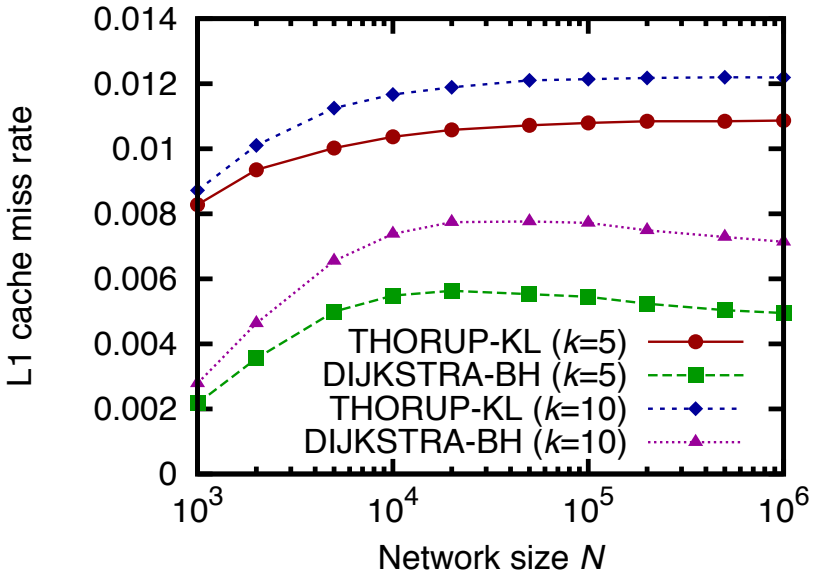


Figure 5.8: L1 cache miss rate of THORUP-KL and DIJKSTRA-BH for the average degree $k = 5, 10$ and multiple edge weights of 1–1000.

BH. Namely, THORUP-KL suffers from much more frequent cache miss penalties than DIJKSTRA-BH, leading significant slow down in program execution.

So, why is the THORUP-KL not cache-friendly? Effectiveness of the memory cache is affected by several factors: the size of memory used and the locality of memory accesses (i.e., spatial and temporal localities) during program execution [82]. As explained in Section 5.2.3, both THORUP-KL and DIJKSTRA-BH are linear space algorithms. Lack of cache-friendliness in THORUP-KL compared with DIJKSTRA-BH can be explained as follows.

- Larger memory usage

  The memory consumption of THORUP-KL is as approximately 1.4 times large as that of DIJKSTRA-BH (see Fig. 5.6). Larger memory

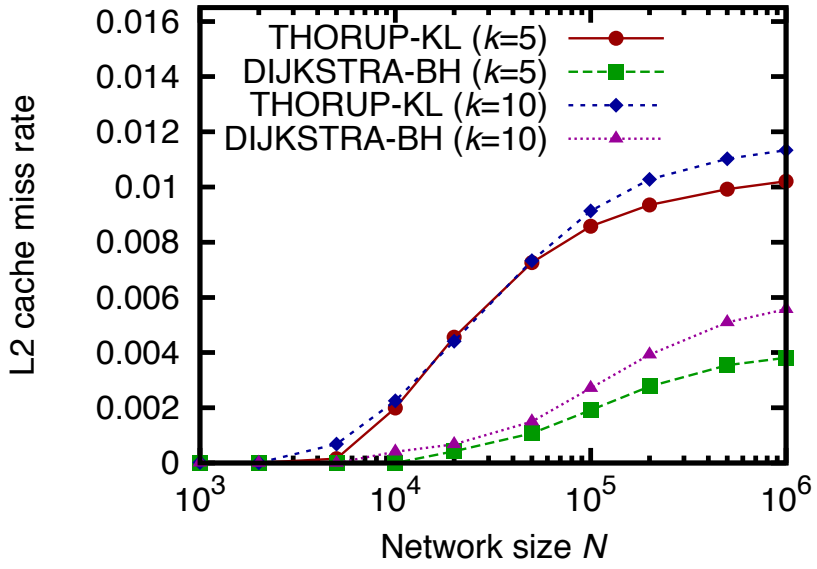


Figure 5.9: L2 cache miss rate of THORUP-KL and DIJKSTRA-BH for the average degree $k = 5, 10$ and multiple edge weights of 1–1000.

consumption results in a higher cache miss rate.

- Lower spatial locality

  The hierarchical bucketing structure of THORUP-KL is sparsely allocated in the memory while the binary heap of DIJKSTRA-BH is densely allocated. The hierarchical bucketing structure consists of several types of elements such as references (i.e., pointers) to buckets and references to child components. The binary heap can be realized as an array. Lower spatial locality of THORUP-KL results in a higher cache miss rate.

- Lower temporal locality

  THORUP-KL needs recursion, whereas DIJKSTRA-BH doesn't. Because of introducing the hierarchical bucketing structure, THORUP-

KL needs to recursively visit components. Recursion generally results in lower temporal locality. On the contrary, DIJKSTRA-BH only requires looping. Lower temporal locality of THORUP-KL results in a higher cache miss rate.

From these observations, we conclude that THORUP-KL is almost always faster than DIJKSTRA-BH for large-scale network simulation. However, as the camel-shaped function of the relative execution time indicates, its effective is significantly affected by the memory cache performance.

5.2.5 Estimating Relative Execution Time

For practically utilizing THORUP-KL for network simulation, thorough understanding of its effective is necessary. In particular, we need to understand how the effectiveness of THORUP-KL is affected by the memory cache performance. Hence, we finally try to answer the last question: how is the relative execution time affected by several factors such as L1 and L2 cache miss rates and their cache miss penalties?

Let us introduce a simple cache performance model [83], which approximates the execution time $\tilde{T}$ of a program on the microprocessor with L1 and L2 memory caches.

$$\tilde{T} = N_I \, T_{CPU} + N_M \, (t_{AL1} + p_{L1} \, T_{L1} + p_{L2} \, T_{L2}) \tag{5.2}$$

In the above equation, N_I is the number of instructions executed, T_{CPU} the CPI (Cycles Per Instruction) time, and N_M the number of memory accesses performed. Also, T_{AL1} is the access time of L1 cache, and T_{Li} and p_{Li} ($i = 1, 2$) are the cache miss penalty and rate of Li cache.

We already have L1 and L2 cache miss rates, p_{L1} and p_{L2}, in Figs. 5.8 and

Table 5.1: Measured CPI time, access time and cache miss penalty

CPI (Cycles Per Instruction) time	T_{CPU}	0.96	[ns]
access time of L1 cache	T_{AL1}	1.4	[ns]
cache miss penalty of L1 cache	T_{L1}	49.4	[ns]
cache miss penalty of L2 cache	T_{L2}	435.3	[ns]

Table 5.2: System specifications

FSB (Front-Side Bus) clock	800	[MHz]
L1 cache size	32	[Kbyte]
L2 cache size	1024	[Kbyte]
memory	DDR2-533	
memory controller hub	Intel 82945G	
memory clock	133	[MHz]
memory bus width	128	[bit]
I/O bus clock	266	[MHz]

5.9. The number of instructions executed, N_I, and the number of memory accesses performed, N_M, can be obtained with cachegrind. We then measured other system-dependent (and program-independent) parameters, T_{CPU}, T_{AL1}, T_{L1}, and T_{L2}, using a benchmark tool called *lmbench* [84]. Our measurement results are summarized in Tab. 5.1.

By substituting all parameters in Eq. (5.2) with our measured values, execution times of THORUP-KL and DIJKSTRA-BH, $\tilde{T_T}$ and $\tilde{T_D}$, can be estimated. Thus, the estimated relative execution time, $\tilde{T_T}/\tilde{T_D}$, can be measured. Figure 5.10 shows the estimated relative execution times obtained from Eq. (5.2) as well as the relative execution time obtained in Section 5.2.2.

Figure 5.10 shows that the simple cache performance model well explains the camel-shaped function of the relative execution time. It should be noted that both the relative execution time and the estimated relative execution time take the maximum value at the same network size N in Fig. 5.10. Namely, the simple cache performance model accurately captures the dynamics of the relative execution time. In other words, the camel-shaped function of the relative execution time can be explained solely with

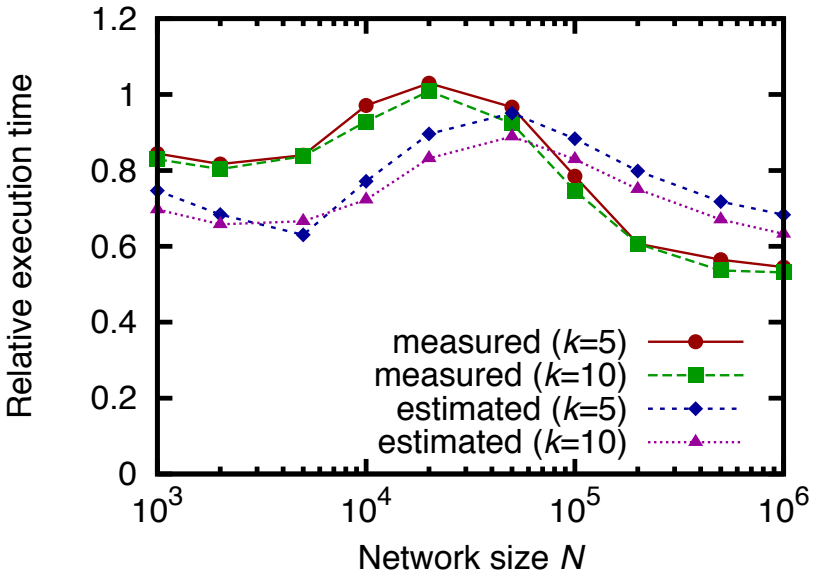


Figure 5.10: Measured and estimated relative execution times with the memory cache for the average degrees $k = 5, 10$ and multiple edge weights of 1–1000.

the memory cache performance.

5.2.6 Discussion

As explained in Section 5.1, the time complexity of THORUP-FR is $O(E + N)$ whereas that of THORUP-KL is $O(E\,\alpha(N) + N)$. We intentionally used THORUP-KL instead of THORUP-FR. Our experiments clearly show that practical efficiency of an algorithm cannot be predicted only from its time complexity. Its performance is significantly affected by the existence of memory cache.

In Sections 5.2.4 and 5.2.5, we have investigated the effect of memory cache on performance of THORUP-KL and DIJKSTRA-BH from two different approaches. It is well known that effectiveness of an algorithm is considerably influenced by memory cache [82, 85, 86]. Also, studies on *cache-*

aware and *cache-oblivious* algorithms have been actively performed [85, 87, 88]. The effect of memory cache on the performance of an algorithm is quite complicated. Therefore, its effectiveness should be carefully investigated with extensive experiments, as we have done in this chapter.

In Section 5.2.2, we have shown that THORUP-KL is almost always faster than DIJKSTRA-BH regardless of the network size, the average degree, and the type of edge weights. This implies that THORUP-KL is advantageous to DIJKSTRA-BH for network simulation. However, as discussed in Section 5.2.1, we have only used a simple network model (i.e., random network). The effectiveness of THORUP-KL and DIJKSTRA-BH might be affected by the type of networks (e.g., hierarchical network and scale-free network). More investigation with realistic network models would be of great interest.

In Section 5.2.5, we have demonstrated the potential of a simple cache performance model for analyzing the effect of system performance on the execution time. For instance, the simple model enables us to predict the effect of system performance improvement/degradation on the execution time. More specifically, the execution time with a double-speed microprocessor can be predicted simply by halving T_{CPU} in Eq. (5.2). Also, the execution time with double-speed L1 cache memory can be predicted simply by halving T_{L1} in Eq. (5.2). Note that program-specific parameters (i.e., the number of instructions executed, N_I, the number of memory accesses performed, N_M, and cache miss rates p_{L1} and P_{L2}) are independent of other system-specific parameters (i.e., T_{CPU}, T_{L1} and T_{L2}). Thus, we can freely change system parameters in Eq. (5.2) to predict their effect on the execution time. We are planning to investigate the effect of system architecture (e.g., type of microprocessors and memory cache architecture) on the effec-

tiveness of THORUP-KL and DIJKSTRA-BH.

It is beyond the scope of this chapter, but investigation on the effectiveness of THORUP-KL and DIJKSTRA-BH in a parallel environment would be of great importance. In this chapter, the performances of THORUP-KL and DIJKSTRA-BH with a single microprocessor are analyzed. However, for very large-scale network simulation, usage of SMP processors and/or parallel computers might be a viable choice. Multiple instances of a single-source shortest path algorithm can be executed in parallel on SMP processors and/or parallel computers.

5.3 Summary

The objective of this chapter is therefore to investigate the effectiveness of Thorup's algorithm by comparing with Dijkstra's, and to answer the following questions.

1. How efficiently/inefficiently does Thorup's algorithm perform compared with Dijkstra's for large-scale (e.g., million nodes) network simulation?

2. How and why does the practical performance of Thorup's and Dijkstra's algorithms deviate from their time complexities (i.e., theoretical performance)?

In this chapter, we have intentionally used THORUP-KL (i.e., a *simplified* version of Thorup's algorithm) with the time complexity of $O(E\,\alpha(N)+N)$.

There are several variants of Dijkstra's algorithm with different time complexities of $O(E+N^2)$ [89], $O((E+N)\,\log N)$ [46] and $O(E+N\,\log N)$ [52].

In this chapter, we have focused on Dijkstra's algorithm with a binary heap (DIJKSTRA-BH) [46] with the time complexity of $O((E + N) \log N)$.

Through extensive experiments with our implementations of THORUP-KL and DIJKSTRA-BH, we have compared their performances (i.e., the execution time and memory consumption). Our findings include that (1) THORUP-KL is almost always faster than DIJKSTRA-BH for large-scale network simulation, and (2) the performances of THORUP-KL and DIJKSTRA-BH deviate from their time complexities due to the existence of memory cache in the microprocessor.

Chapter 6

Conclusion

In this chapter, we summarize the research presented in this thesis, and present directions for future works.

In Chapter 2, we have analyzed the stability of XCP in a network with heterogeneous XCP flows (i.e., XCP flows with different propagation delays). Through several numerical examples and simulation results, we have investigated the effect of system parameters and XCP control parameters on stability of the XCP protocol. Our findings include: (1) when XCP flows are heterogeneous, XCP operates more stably than the case when XCP flows are homogeneous, (2) conversely, when variation in propagation delays of XCP flows is large, operation of XCP becomes unstable, and (3) the output link bandwidth of an XCP router is independent of stability of the XCP protocol.

In Chapter 3, we have proposed XCP-IR (XCP with Increased Robustness) that operates efficiently even for dynamic traffic. XCP-IR prevents instability of the XCP control caused by non-XCP traffic dynamics while preventing degradation of the bottleneck-link utilization caused by XCP traffic dynamics. We have analyzed stability and transient state performance of

XCP-IR using the analytic approach in Chapter 2. Through extensive simulation experiments, we have shown that XCP-IR operated efficiently even for dynamic traffic. In particular, we have shown that the throughput with XCP-IR is approximately 200% higher than that with XCP.

In Chapter 4, we have proposed a flow-level simulator called FSIM (Fluid-based SIMulator) for performance evaluation of large-scale networks, and have verified its effectiveness using our FSIM implementation. Through extensive experiments using our FSIM implementation, we have evaluated the effectiveness of our flow-level simulator FSIM in terms of simulation speed, accuracy and memory consumption. We have shown that our flow-level simulator FSIM outperforms a conventional flow-level simulator; i.e., it realizes approximately 200%-300% faster simulation with higher accuracy and less memory consumption than a conventional flow-level simulator. In particular, we should note that FSIM is effective for performance evaluation of a network with large link capacities and many TCP flows.

The objective of Chapter 5 is to investigate the effectiveness of Thorup's algorithm by comparing with Dijkstra's, and to answer the following questions.

1. How efficiently/inefficiently does Thorup's algorithm perform compared with Dijkstra's for large-scale (e.g., million nodes) network simulation?

2. How and why does the practical performance of Thorup's and Dijkstra's algorithms deviate from their time complexities (i.e., theoretical performance)?

In Chapter 5, we have intentionally used THORUP-KL (i.e., a *simplified* version of Thorup's algorithm) with the time complexity of $O(E\,\alpha(N)+N)$.

There are several variants of Dijkstra's algorithm with different time complexities of $O(E+N^2)$ [89], $O((E+N)\ \log N)$ [46] and $O(E+N\ \log N)$ [52]. In this thesis, we have focused on Dijkstra's algorithm with a binary heap (DIJKSTRA-BH) [46] with the time complexity of $O((E+N)\ \log N)$.

Through extensive experiments with our implementations of THORUP-KL and DIJKSTRA-BH, we have compared their performances (i.e., the execution time and memory consumption). Our findings include that (1) THORUP-KL is almost always faster than DIJKSTRA-BH for large-scale network simulation, and (2) the performances of THORUP-KL and DIJKSTRA-BH deviate from their time complexities due to the existence of memory cache in the microprocessor.

In this thesis, to realize analysis and simulation of Internet congestion control mechanisms for large-scale networks, we have tackled the issues with fluid-based approaches. For realizing analysis and simulation of Internet congestion control mechanisms for large-scale networks, we are planning to investigate the effectiveness of analytic approach in Chapter 2 in large-scale network with many nodes. We are planning to accelerate FSIM implemented in Chapter 4 by using parallel computation. Moreover, we are planning to improve cache performance of Thorup's algorithm for large scale network simulations.

Bibliography

[1] "Global internet geography." available at http://www.telegeography.com/.

[2] G. Gilder, *TELECOSM: How infinite bandwith will revolutionize our world*. Free Press, 2000.

[3] "ISC internet domain survey." http://www.isc.org/ops/ds/.

[4] S. Floyd and V. Paxson, "Why we don't know how to simulate the Internet," Oct. 1999. available at http://www.aciri.org/floyd/papers/wsc.ps.

[5] F. Cen, T. Xing, and K.-T. Wu, "Real-time performance evaluation of line topology switched ethernet," *International Journal of Automation and Computing*, vol. 5, no. 4, pp. 376–380, 2008.

[6] T. Miyachi, K. ichi Chinen, and Y. Shinoda, "Starbed and springos: large-scale general purpose network testbed and supporting software," in *Proceedings of the 1st international conference on Performance evaluation methodolgies and tools*, p. 30, Oct. 2006.

[7] P. Defibaugh-Chavez, S. Mukkamala, and A. H. Sung, "Efficacy of coordinated distributed multiple attacks (a proactive approach to cyber

defense)," in *Proceedings of the 20th International Conference on Advanced Information Networking and Applications-Volume 02*, pp. 10–14, 2006.

[8] P. Velho and A. Legrand, "Accuracy study and improvement of network simulation in the simgrid framework," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, p. 13, Mar. 2009.

[9] R. Jain, *The Art of computer systems performance analysis*. Wiley - Interscience, Apr. 1991.

[10] R. Pan and B. Prabhakar, "SHRiNK: A method for enabling scaleable performance prediction adn efficient network simulation," in *Proceedings of IEEE GLOBECOM 2005*, pp. 1108–1113, June 2005.

[11] A. Feketea, G. Vattaya, and L. Kocarev, "Traffic dynamics in scale-free networks," *Complex Systems*, vol. 3, pp. 97–107, Aug. 2006.

[12] C. Kiddle, R. Simmonds, and B. Unger, "Improving scalability of network emulation through parallelism and abstraction," in *Proceedings of IEEE ANSS 2005*, pp. 119–129, Apr. 2005.

[13] "The network simulator – ns-2." available at `http://www.isi.edu/nsnam/ns/`.

[14] Y. GU, Y. Lie, and D. Towsley, "On integragin fluid model with packet simulation," in *Proceedings of IEEE INFOCOM 2004*, vol. 4, pp. 2856–2866, Mar. 2004.

[15] Y. Zhang and M. Ahmed, "A control theoretic analysis of XCP," in *Proceedings of IEEE INFOCOM 2005*, vol. 4, pp. 2831–2835, Mar. 2005.

[16] S. H. Low, L. L. H. Andrew, and B. P. Wydrowski, "Understanding XCP: Equilibrium and fairness," in *Proceedings of IEEE INFOCOM 2005*, vol. 2, pp. 1025–1036, Mar. 2005.

[17] D. X. Wei, C. Jin, S. H. Low, and S. Hegde, "FAST TCP: Motivation, architecture, algorithms, performance," *IEEE/ACM Transactions on Networking*, pp. 1–14, 2007.

[18] S. H. Low, F. Paganini, J. Wang, and J. C. Doyle, "Linear stability of TCP/RED and a scalable control," *Computer Networks Journal*, vol. 43, pp. 633–647, Dec. 2003.

[19] J. Postel, "Transmission control protocol," *Request for Comments (RFC) 793*, Sept. 1981.

[20] J. Padhye and S. Floyd, "On inferring TCP behavior," *ACM SIGCOMM Computer Communication Review*, vol. 31, pp. 287–298, Aug. 2001.

[21] S. Floyd, "Highspeed TCP for large congestion windows," *Request for Comments (RFC) 3649*, Dec. 2003.

[22] H. Bullot, R. L. Cottrell, and R. Hughes-Jones, "Evaluation of advanced TCP stacks on fast long-distance production networks," in *Proceedings of PFLDnet 2004*, Feb. 2004.

[23] R. Wang, G. Pau, K. Yamada, M.Y.Sanadidi, and M. Geria, "TCP startup performance in large bandwidth delay networks," in *Proceedings of IEEE INFOCOM 2004*, vol. 2, pp. 796–805, Mar. 2004.

[24] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *Proceedings of ACM SIGCOMM 2002*, vol. 32, pp. 89–102, Aug. 2002.

[25] K. Ramakrishnan, S. Floyd, and D. B. Rosen, "The addition of explicit congestion notification (ECN) to IP," *Request for Comments (RFC) 3168*, Sept. 2001.

[26] M. Welzl, "Scalable router aided congestion avoidance for bulk data transfer in high speed networks," in *Proceedings of PFLDnet2005*, Feb. 2005.

[27] Y. Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman, "One more bit is enough," in *Proceedings of ACM SIGCOMM 2005*, pp. 37–48, Aug. 2005.

[28] A. Falk and D. Katabi, "Specificaton for the explicit control protocol(XCP)," *IETF Internet Draft: draft-falk-xcp-spec-01.txt*, Oct. 2005.

[29] D. M. Lopez-Pacheco, C. Pham, and L. Lefévre, "XCP-i : explicit control protocol for heterogeneous inter-networking of high-speed networks," in *Proceedings of 49th IEEE Global Telecommunications Conference(GLOBECOM 2006)*, pp. 1–6, Nov. 2006.

[30] H. Balakrishnan, N. Dukkipati, N. McKeown, and C. Tomlin, "Stability analysis of explicit congestion control protocols," *IEEE Communications Letters*, vol. 11, pp. 823–825, Oct. 2007.

[31] B. Liu, D. R. Figueired, Y. Guo, J. Kurose, and D. Towsley, "A study of networks simulation efficiency: Fluid simulation vs. packet-level simulation," in *Proceedings of IEEE INFOCOM 2001*, vol. 3, pp. 22–26, Jan. 2001.

[32] Y. Sakumoto, H. Ohsaki, and M. Imase, "Stability analysis of transport protocol XCP for high-speed networks," *Technical Report of IEICE* (IN2006-29), pp. 73–78, June 2006.

[33] Y. Sakumoto, H. Ohsaki, and M. Imase, "On XCP stability in a heterogeneous network," in *Proceedings of 12th IEEE Symposium on Computers and Communications(ISCC'07)*, pp. 531–537, July 2007.

[34] Y. Sakumoto, H. Ohsaki, and M. Imase, "Stability analysis of XCP (eXplicit Control Protocol) with heterogeneous flows," *IEICE Transactions on Communications*, vol. E92-B, pp. 3174–3182, Oct. 2009.

[35] Y. Sakumoto, H. Ohsaki, and M. Imase, "Proposal of a technique for improving robustness of data transfer protocol XCP," *Technical Report of IEICE* (IN2006-91), pp. 13–18, Nov. 2006.

[36] Y. Sakumoto, H. Ohsaki, and M. Imase, "Increasing robustness of XCP (eXplicit Control Protocol) for dynamic traffic," in *Proceedings of 50th IEEE Global Telecommunications Conference(GLOBECOM 2007)*, pp. 2025–2030, Nov. 2007.

[37] Y. Sakumoto, H. Ohsaki, and M. Imase, "Improving robustness of XCP (eXplicit Control Protocol) for dynamic traffic," submitted to *IEICE Transactions on Communications*, Oct. 2009.

[38] Y. Sakumoto, R. Asai, H. Ohsaki, and M. Imase, "Design and implementation of flow-level simulator for performance evaluation of large scale networks," in *Proceedings of 15th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems(MASCOTS) 2007*, pp. 166–172, Oct. 2007.

[39] Y. Liu, F. L. Presti, V. Misra, D. Towsley, and Y. Gu, "Fluid models and solutions for large-scale IP networks," in *Proceedings of ACM/SIGMETRICS 2003*, pp. 91–101, June 2003.

[40] H. Ohsaki, J. Ujiie, and M. Imase, "On scalable modeling of TCP congestion control mechanism for large-scale IP networks," in *Proceedings of IEEE SAINT 2005*, pp. 361–369, Feb. 2005.

[41] Y. Sakumoto, H. Ohsaki, and M. Imase, "On the effectiveness of thorup's shortest path algorithm for large-scale network simulation," to be presented at *the First Workshop on High Speed Network and Computing Environments for Scientific Applications (HSNCE 2010)*, July 2010.

[42] Y. Sakumoto, H. Ohsaki, and M. Imase, "Effectiveness of thorup's shortest path algorithm for large-scale network simulation," submitted to *IEICE Transactions on Communications*, June 2010.

[43] D. M. Nicol, M. Liljenstam, and J. Liu, "Advanced concepts in large-scale network simulation," in *Proceedings of the 37th conference on Winter simulation*, pp. 153–166, Dec. 2005.

[44] P. Huang and J. Heidemann", "Minimizing routing state for lightweight network simulation," in *Proceedings of The IEEE Computer Society's 9th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems(MASCOTS)*, pp. 108–116, Aug. 2001.

[45] Z. Hao, X. Yun, and H. Zhang, "An efficient routing mechanism in network simulation," *Simulation*, vol. 84, pp. 511–520, May 2008.

[46] J. Williams, "Heapsort," *Communications of the ACM*, vol. 7, no. 6, pp. 347–348, 1964.

[47] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms (3rd ed.)*. MIT Press, Sept. 2009.

[48] M. Thorup, "Undirected single-source shortest paths with positive integer weights in linear time," *Journal of the ACM (JACM)*, vol. 46, pp. 362–394, May 1999.

[49] M. L. Fredman and D. E. Willard, "Trans-dichotomous algorithms for minimum spanning trees and shortest paths," *Journal of Computer and System Sciences*, vol. 48, pp. 533–551, June 1994.

[50] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," in *Proceedings of the American Mathematical society*, pp. 48–50, JSTOR, Feb. 1956.

[51] Y. Asano and H. Imai, "Practical efficiency of the linear-time algorithm for the single source shortest path problem," *Journal of the Operations Research Society of Japan*, vol. 43, pp. 431–447, Dec. 2000.

[52] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *Journal of the ACM (JACM)*, vol. 34, pp. 596–615, July 1987.

[53] Y. Zhang, D. Leonard, and D. Loguinov, "Jetmax: Scalable max-min congestion control for high-speed heterogeneous networks," in *Proceedings of IEEE INFOCOM 2006*, pp. 1–13, Apr. 2006.

[54] H. Hisamatsu, H. Ohsaki, and M. Murata, "Fluid-based analysis of network with DCCP connections and RED routers," in *Proceedings of IEEE SAINT 2006*, pp. 156–163, Jan. 2006.

[55] N. S. Nise, *Control Systems Engineering*. New York: John Wiley & Sons, 4th ed., Aug. 2003.

[56] A. Falk, Y. Pryadkin, and D. Katabi, "Specificaton for the explicit control protocol(XCP)," *IETF Internet Draft: draft-falk-xcp-spec-03.txt*, July 2007.

[57] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," *Request for Comments (RFC) 2581*, Apr. 1999.

[58] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A compound TCP approach for high-speed and long distance networks," in *Proceedings of 25th IEEE International Conference on Computer Communications(INFOCOM 2006)*, pp. 1–12, Apr. 2006.

[59] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 397–413, Aug. 1993.

[60] J. Aweya, M. Ouellette, and D. Y. Montuno, "Service differentiation using a multi-level red mechanism," *International Journal of Network Management*, vol. 12, pp. 81–98, Jan. 2002.

[61] T. Mori, M. Uchida, and S. Goto, "Flow analysis of internet traffic: world wide web versus peer-to-peer," *Systems and Computers in Japan*, vol. 36, pp. 70–81, Oct. 2005.

[62] A. Kavimandan, W. Lee, M. T. A. Gokhale, and R. Viswanathan, "Netowrk simulation via hybrid system modeling: A time-stepped approach," in *Proceedings of ICCCN 2005*, pp. 531–536, Oct. 2005.

[63] J. Liu, "A primer for real-time simulation of large-scale networks," in *Proceedings of the 41st Annual Simulation Symposium (ANSS'08)*, pp. 85–94, Apr. 2008.

[64] J. Zhou, Z. Ji, M. Takai, and R. Bagrodia, "Maya: a multi-paradigm network modeling framework for emulating disributed applications," in *Proceedings of IEEE PADS 2003*, pp. 162–170, June 2003.

[65] B. Melamed, S. Pan, and Y. Wardi, "Hybrid discrete-continuous fluid-flow simulation," in *Proceedings of IEEE SPIE 2001*, vol. 4526, pp. 263–270, July 2001.

[66] J. R. Dormand and P. J. Prince, "A family of embedded runge-kutta formulae," *Journal of Computational and Applied Mathematics*, vol. 6, pp. 19–26, Mar. 1980.

[67] E. Hairer, S. P. Norsett, and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer-Verlag, 1993.

[68] "Otcl(mit object tcl)." `http://otcl-tclcl.sourceforge.net/otcl/`.

[69] "The network animator – NAM." available at `http://www.isi.edu/nsnam/nam/`.

[70] "Simulating large networks using fluid flow models (FFM)." available at `http://www-net.cs.umass.edu/fluid/fluid.html`.

[71] L. Andrew, S. Floyd, and G. Wang, "Common tcp evaluation suite," *IETF Internet Draft: draft-irtf-tmrg-tests-02.txt*, July 2009.

[72] B. Bloolbas, *Random Graphs Second Edition*. Cambridge Univ Press, Oct. 2001.

[73] O. Hiroyuki, Y. Koutaro, and I. Makoto, "On the effect of scale-free structure of network topology on t cp performance," *SAINT '07:proceeding of the 2007 International Symposium on Ap plications and the Internet*, p. 12, 2007.

[74] G. Peli and G. Papp, "Are scale free networks better?." `http://arxiv.org/pdf/cond-mat/0301555`, Feb. 2003.

[75] M. E. J. Newman, S. H. Strogatz, and D. J. Watts, "Random graphs with arbitrary degree distributions and their applications," *Phycal Review E*, vol. 64, p. 026118, July 2001.

[76] K. Calvert, M. B. Doar, A. Nexion, E. W. Zegura, G. Tech, and G. Tech, "Modeling internet topology," *IEEE Communications Magazine*, vol. 35, pp. 160–163, June 1997.

[77] "The network simulator – ns2: Tips and statistical data for running large simulations in ns." `http://www.isi.edu/nsnam/ns/ns-largesim.html`.

[78] H. Hisamatu, H. Ohsaki, and M. Murata, "Fluid-based analysis of a network with DCCP connections and RED routers," in *Proceedings of the 2006 International Symposium on Applications and the Internet (SAINT 2006)*, pp. 22–29, Jan. 2006.

[79] H. Ohsaki, H. Yamamoto, and M. Imase, "Scalable modeling and performance evaluation of dynamic RED router using fluid-flow approximation," in *Proceedings of OpticsEast/ITCom 2005*, Oct. 2005.

[80] H. Tangmunarunkfit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "Network topology generators: Degree-based vs. structural," *ACM SIGCOMM Computer Communication Review*, vol. 32, pp. 147–159, Oct. 2002.

[81] "Cachegrind - a cache profiler." `http://valgrind.org/info/tools.html`.

[82] J. Handy, *The cache memory book*. Morgan Kaufmann, Jan. 1998.

[83] J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach, 3rd Edition*. Morgan Kaufmann, May 2002.

[84] "Lmbench - tools for performance analysis." `http://www.bitmover.com/lmbench/`.

[85] R. E. Ladner, R. Fortna, and B.-H. Nguyen, "A comparison of cache aware and cache oblivious static search trees using program instrumentation," *Experimental algorithmics*, pp. 78–92, Jan. 2002.

[86] A. LaMarca and R. E. Ladner, "The influence of caches on the performance of sorting," in *Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*, pp. 370–379, Jan. 1997.

[87] G. S. Brodal, R. Fagerberg, and R. Jacob, "Cache oblivious search trees via binary trees of small height," in *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 39–48, Jan. 2002.

[88] L. Arge, M. A.Bender, E. D.Demaine, B. Holland-Minkley, and J. Munro, "Cache-oblivious priority queue and graph algorithm applications," in *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pp. 268–276, May 2002.

[89] E. W. DijkstraDijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, pp. 269–271, Dec. 1959.