Fall 1-31-2006

Network level performance of differentiated services (diffserv) networks

Li Zhu
New Jersey Institute of Technology

ABSTRACT

NETWORK LEVEL PERFORMANCE OF DIFFERENTIATED SERVICES (DIFFSERV) NETWORKS

by
Li Zhu

The Differentiated Services (DiffServ) architecture is a promising means of providing Quality of Service (QoS) in Internet. In DiffServ networks, three service classes, or Per-hop Behaviors (PHBs), have been defined: Expedited Forwarding (EF), Assured Forwarding (AF) and Best Effort (BE).

In this dissertation, the performance of DiffServ networks at the network level, such as end-to-end QoS, network stability, and fairness of bandwidth allocation over the entire network have been extensively investigated.

It has been shown in literature that the end-to-end delay of EF traffic can go to infinity even in an over-provisioned network. In this dissertation, a simple scalable aggregate scheduling scheme, called Youngest Serve First (YSF) algorithm is proposed. YSF is not only able to guarantee finite end-to-end delay, but also to keep a low scheduling complexity.

With respect to the Best Effort traffic, Random Exponential Marking (REM), an existing AQM scheme is studied under a new continuous time model, and its local stable condition is presented. Next, a novel virtual queue and rate based AQM scheme (VQR) is proposed, and its local stability condition has been presented. Then, a new AQM framework, Edge-based AQM (EAQM) is proposed. EAQM is easier to implement, and it achieves similar or better performance than traditional AQM schemes.

With respect to the Assured Forwarding, a network-assist packet marking (NPM) scheme has been proposed. It has been demonstrated that NPM can fairly distribute bandwidth among AF aggregates based on their Committed Information Rates (CIRs) in both single and multiple bottleneck link networks.

HighSpeed TCP (HSTCP) provides reliable data transmission at very high speed over Internet. At the same time, Optical Burst Switching (OBS) becomes a promising technology to support the future Internet backbone. A simple model for a single HSTCP connection over an OBS network is proposed to investigate the impact of OBS on the throughput of a single HSTCP connection.

NETWORK LEVEL PERFORMANCE OF DIFFERENTIATED SERVICES (DIFFSERV) NETWORKS

by
Li Zhu

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Electrical Engineering

Department of Electrical and Computer Engineering

January 2006

APPROVAL PAGE

NETWORK LEVEL PERFORMANCE OF DIFFERENTIATED SERVICES (DIFFSERV) NETWORKS

Li Zhu

Dr. Nirwan Ansari, Dissertation Advisor — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Aleksandar Kolarov, Committee Member — Date
Technical Leader, Cisco Systems Inc.

Dr. Roberto Rojas-Cessa, Committee Member — Date
Assistant Professor of Electrical and Computer Engineering, NJIT

Dr. Sirin Tekinay, Committee Member — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Meng-Chu Zhou, Committee Member — Date
Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Li Zhu

Degree: Doctor of Philosophy

Date: January 2006

Undergraduate and Graduate Education:

- Doctor of Philosophy in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2006
- Master of Science in Electrical Engineering,
  Georgia Institute of Technology, Atlanta, GA, 2001
- Master of Science in Physics,
  Nanjing University, Nanjing, Jiangsu, China, 1997
- Bachelor of Science in Physics,
  Nanjing University, Nanjing, Jiangsu, China, 1994

Major: Electrical Engineering

Presentations and Publications:

L. Zhu, G. Cheng, K. Xu and N. Ansari, "Edge-based Active Queue Management (EAQM)," accepted by *IEE Proc. Communications*.

G. Cheng, L. Zhu and N. Ansari, "A New Deterministic Traffic Model for Core-stateless Scheduling," accepted by *IEEE Transactions on Communications*.

L. Zhu, N. Ansari, J. Liu, "Throughput of HighSpeed TCP in Optical Burst Switching Networks," *IEE Proc. Communications*, vol. 152, no. 3, pp. 349–352, June 2005.

L. Zhu, N. Ansari, "Fair Bandwidth Allocation for Assure Forwarding (AF) Services," *IEEE 2005 International Conference on Communications*, (*ICC 2005*), Seoul, May 2005, vol. 1, pp. 374–378.

G. Cheng, N. Ansari, and L. Zhu, "Framework for Finding the Optimal Linear Scaling Factor of Epsilon-approximation Solutions," *IEEE 2005 International Conference on Communications*, (*ICC 2005*), Seoul, May 2005, vol. 2, pp. 866–870.

L. Zhu and N. Ansari, "Local Stability of A New Adaptive Queue Management (AQM) Scheme," *IEEE Communication Letters*, vol. 8, no. 6, pp. 406–408, June 2004.

G. Cheng, L. Zhu and N. Ansari, "A New Traffic Model for Core-stateless Scheduling," *IEEE 2003 Global Communications Conference* (*GLOBECOM 2003*), San Francisco, CA, USA, December 2003, vol. 6, pp. 3711–3715.

L. Zhu, G. Cheng and N. Ansari, "Local stable condition for random exponential marking," *IEE Proc. Communications*, vol. 150, no. 5, pp. 367–370, October 2003.

L. Zhu, N. Ansari, Z. Sahinoglu, A. Vetro, and H. Sun, "Scalable Layered Multicast with Explicit Congestion Notification," *2003 International Conference on Information Technology: Coding and Computing* (*ITCC 2003*), Las Vegas, Nevada, USA, April 2003, pp. 331–335.

L. Zhu, G. Cheng, N. Ansari, Z. Sahinoglu, A. Vetro, and H. Sun, "On Proxy Caching for Video on Demand Systems in Multicasting Networks," *2003 Conference on Information Sciences and Systems* (*CISS 2003*), Johns Hopkins University, Maryland, USA, March 2003.

L. Zhu, G. Cheng and N. Ansari, "On The Delay Bound of Youngest Serve First Aggregated Packet Scheduling," *IEE Proc. Communications*, vol. 150, no. 1, pp. 6–10, February 2003.

To my parents and wife.

ACKNOWLEDGMENT

First and foremost, I would like to thank my dissertation adviser, Professor Nirwan Ansari for giving me the chance to work with him in the the past four and half years. Professor Ansari not only guided and helped me on the research, but also showed me that being a humble and decent person is more important than a great researcher.

I also want to thank all other members in my dissertation committee, Dr. Aleksandar Kolarov, Dr. Symeon Papavassiliou, Dr. Roberto Rojas-Cessa, Dr. Sirin Tekinay and Dr. Meng-Chu Zhou, for their invaluable discussions and suggestions. This dissertation would not be possible without their generosity and support.

I would like to thank all my friends and colleagues in the Advanced Networking Lab (ANL). Special thanks go to Gang Cheng, Ye Tian and Kai Xu for their friendship and help.

Most especially, I would like to thank my parents for their unconditional love and support. I also want to thank my wife for sharing the best life I ever can imagine.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter Page

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

The current Internet only supports Best Effort (BE) service, in which all users' traffics are treated in the same way, regardless of their application types and special service requirements. This service model worked well in the early years of the Internet since the majority of the traffics were web browsing, FTP, and emails at that time. Today, with new emerging applications, such as multimedia streaming and VoIP, there is a big demand for a new service model that can provide different services to different applications based on their quality of service (QoS) requirements, e.g., delay, delay jitter and packet dropping rate.

Internet Engineering Task Force (IETF) has proposed two service models, namely Integrated Services (IntServ) [1] and Differentiated Services (DiffServ) [2] to provide QoS in the Internet. With call administration control and bandwidth reservation, IntServ provides end-to-end QoS guarantees for different types of applications. However, lacking scalability and flexibility brings tremendous technical difficulty to implement IntServ in the Internet. DiffServ tries to provide QoS in a much coarser level, or namely in the aggregate level instead of providing QoS to each session. In DiffServ, every packet is classified into a small number of Per Hop Behavior (PHB) aggregates, such as Expedite Forwarding (EF), Assured Forwarding (AF), and BE at network edges. Packets in the same aggregates are treated in the same way in the core of the networks.

1.1 Integrated Services

The IntServ model is characterized by resource reservation and admission control. Before each application session starts, a path is set up and a certain amount of resources (bandwidth, buffer, etc.) are reserved along the path in advance; after the session finishes,

the reserved resource is released. There are four building blocks in the IntServ model: call admission control, bandwidth reservation, classifier, and packet scheduler [1].

1. *Call Admission Control* — The call admission control unit is used to determine if a new application session can be provided with its desired QoS without affecting the QoS of the existing sessions. If there is not enough bandwidth to accept this new session, it is either rejected or its QoS is downgraded.

2. *Bandwidth Reservation* — In order to guarantee QoS, a certain amount of resources have to be reserved for each session. RSVP [3] is the signaling protocol to reserve resources. When a new session arrives, the sender sends the receiver the *Path* message, which specifies its QoS requirement. Upon receiving the *Path* message, the receiver sends back the *Resv* message, which carries the resource reservation request to the routers along the path from the sender to the receiver. Each router along the path decides if the *Resv* request can be accepted. Once the request is accepted by all the routers along the path, necessary resources are reserved. If some of the routers cannot grant the resources requested, the session is either rejected or the network provides a lower QoS.

3. *Classifier* — Before packets enter the network, they are classified into different classes so that they receive different treatment at the packet scheduler. A class could be a particular application session or multiple flows.

4. *Packet Scheduler* — There are several scheduling schemes that can be adopted by the packet scheduler: priority queuing, round robin, Weighted Fair Queuing (WFQ), etc.. Generally speaking, each queue corresponds to one class. If some classes consist of a single session each, a huge number of queues are to be maintained by each router, and this is very difficult to implement.

In addition to the BE service, IntServ proposed two service classes: Guarantee Service [4] and Controlled-Load Service [5]. Guarantee Service aims to provide hard delay bound

and no packet loss for the conforming traffics, and it is intended for applications such as real time audio and video delivery, which need strict delay guarantee. Each flow's traffic is characterized in the form of a token bucket, a minimum policed unit (m) and a maximum datagram size (M). At the network edge, Guarantee Service flow's traffic is shaped according to its traffic parameters. The required amount of reserved bandwidth and buffer are computed based on the fluid model to guarantee the application's QoS [6]. The Controlled-Load Service does not intend to provide strict QoS such as delay bound. Instead, it provides service equivalent to that received by the BE traffic on a slightly loaded network. Such type of service is suitable for adaptive real time applications.

Although IntServ is able to provide QoS to each flow, the lack of scalability and flexibility prevents it from practical deployment. IntServ is a flow oriented service model, in which routers have to maintain state information for each active flow and perform packet checking and scheduling at the flow level. It is extremely difficult to implement IntServ on the major ISP backbones, on which a large number of active flows exist and millions of packets are processed every second. In order to overcome these disadvantages, IETF took a fresh approach, DiffServ, which will be introduced in the next section.

1.2 Differentiated Services

The philosophy of DiffServ is two fold: first, keeping the network core as simple as possible and pushing the complexity to the network edge; second, providing service differentiation at the per aggregate level instead of per flow level. By doing so, DiffServ is more scalable, manageable, and implementable.

DiffServ defines three Per-hop Behaviors (PHBs): Expedite Forwarding (EF), Assured Forwarding (AF), and Best Effort (BE), and each PHB represents one service level. At network edges, packets are classified into these classes and the corresponding PHB is identified by Differentiated Service Code Point (DSCP) in the Type of Service

(TOS) field in the IP headers. At the network core, each packet is processed only based on the DSCP it carries.

DiffServ has great advantages over IntServ. First, there are only limited number of service classes (PHBs), and the core routers only need to manage the aggregate flows. Therefore, it becomes feasible to implement packet scheduling because small number of queues are maintained. Second, packet classification and conditioning are performed only at the network edges; this greatly reduces the processing overhead at core nodes and makes DiffServ deployable at ISP backbones.

DiffServ also has some disadvantages. First, DiffServ provides coarse QoS granularity at the aggregate level, and does not guarantee the QoS for individual flows in each aggregate. Second, PHBs only have local meaning at each hop; it is unclear what kind of end-to-end QoS can be provided.

1.2.1 Expedite Forwarding

The EF class is intended for delay and loss sensitive applications such as Voice over IP (VoIP). One approach to implement EF is to use priority queueing by giving the highest priority to the EF traffic. Another approach is to use WFQ by offering the EF class a much larger weight as compared to other classes. The EF queue adopts the simplest DropTail queue management scheme. In order to achieve low delay, the queuing delay at EF queue has to be maintained at a certain low level. However, traffic burstiness can cause high queuing delay, which is undesirable for applications served in the EF class. Three approaches can be used to alleviate the effect of burstiness on queuing delay. First, ingress nodes perform traffic shaping so that the traffic conforms to certain traffic envelope and it becomes less bursty. Second, the queuing delay can be reduced by over-provisioning bandwidth for the EF class. Third, networks perform admission control for the EF traffics.

1.2.2 Assured Forwarding

The AF class is suitable for applications that requires more assured service than BE even when network congestion happens, and video streaming is one of such applications. In the AF model, customers and ISPs have Service Level Agreements (SLAs), which specify the traffic profiles of customers. The traffics within the profiles are called "IN" traffics, and those out of profile are called "OUT" traffics. Both "IN" and "OUT" packets are served in First In First Out (FIFO) manner. Since there is no admission control for AF, the network bandwidth can be either over-provisioned or under-provisioned. When excess bandwidth is available, both "IN" and "OUT" packets are forwarded. When congestion happens, "IN" packets are served with higher priority by dropping "OUT" packets with higher probability.

AF can be realized with random early detection (RED) with In and Out (RIO), which is also called the two-color marking scheme. RIO adopts the two-level RED scheme, in which there are two thresholds min_th and max_th. When the queue length is less than min_th, all packets are forwarded. When the queue length is between min_th and max_th, OUT packets are dropped with a certain probability and IN packets are forwarded without dropping. When the queue length is larger than max_th, IN packets are dropped based on certain probabilities and all OUT packets are dropped. Three-color marking adopts the similar approach to RIO, except that three-level RED is used and traffics are characterized into three levels instead of two in RIO.

1.3 Congestion Control, TCP and Active Queue Management (AQM)

Internet has finite resources (bandwidth and buffer). When total user demands exceed the available resources, congestion happens [8]. Network congestion can cause larger queuing delay, less effective bandwidth utilization and even congestion collapse [7]. From the network users' perspective, the received services will be severely downgraded and unacceptable. Therefore, congestion control plays a very important role in keeping good network performance and providing good quality services to users.

There are two key components in congestion control: transmission protocols at end users and active queue management at network routers. TCP is the most dominant transport protocol adopted by end users. It was reported that about 95% of Internet traffics are generated from TCP connections [9]. TCP decreases its tranmission rate when congestion is detected, and increases it when spare bandwidth is available. DropTail is the simplest and still the most dominant queue management scheme in Internet. In the DropTail scheme, packets are dropped only when the buffer is overflowed. Upon sensing the packet dropping, TCP decreases its transmission rate. Then, TCP increases its transmission rate slowly till congestion happens again. However, DropTail has unavoidable disadvantages. First, many of TCP connections may decrease and increase their transmission rates at the same time; this phenomenon is called TCP synchronization, which can signficantly reduce bandwidth utilization. Second, the queue length may oscillate between zero and the buffer size, and this can cause undesired queuing delay and delay jitter. Third, not the last, the combination of TCP and DropTail is a reactive congestion control, which is triggered only when severe congestion happens.

Active Queue Management (AQM) aims to overcome the disadvantages of Drop-Tail. In AQM, routers measure the congestion level on their links and signal the congestion condition explicitly to users by probabilistically dropping or marking packets. Many AQM schemes have been proposed, e.g., random early detection (RED) [13], random exponential marking (REM) [14], PI controller [15], adaptive virtual queue (AVQ) [16], and state feedback control (SFC) [18]. These AQM proposals differ in the ways they measure congestion level and the dropping probability is computed. However, the goals of these schemes are similar: providing high link utilization, eliminating TCP synchronization, and stabilizing queue length.

1.3.1 TCP

TCP is a congestion responsive protocol, which dynamically adapts its transmission rate according to the congestion condition. Based on the way it detects congestion and how it responds to it, TCP can be categorized into two types: packet dropping (marking) based and queuing delay based. The first category includes Reno [19], New Reno [20], SACK [21] and High-Speed-TCP (HSTCP) [22]. The second category includes Vegas [23] and FAST [24].

Reno and NewReno are deployed by a majority of current Internet users. These protocols adopt Additive Increase Multiplicative Decrease (AIMD) window adaptation, in which the window size is increased by one every round trip time (RTT) if no packet is dropped, and the window size is halved when congestion (indicated by packet dropping) is detected. This AIMD scheme works well in low bandwidth and small delay networks, and is not suitable for networks with high bandwidth and long delay. In high bandwidth and long delay networks, AIMD drops the window size too much when congestion happens, and it takes very long time to fully utilize bandwidth again since the window size is increased only by one each RTT. In order to address this issue, HSTCP was proposed. HSTCP is still an AIMD scheme, but it implements more conservative window dropping and more aggressive window increasing scheme when bandwidth is high. As a result, HSTCP can make much better use of bandwidth.

Packet dropping based TCP has some unavoidable disadvantages. First, dropping window size multiplicatively decreases TCP transmission rate too drastically. As a result, the link utilization may not be high and the queue size can oscillate greatly. Second, packet dropping only provides one bit information about congestion: congestion either happens or not [24]. If more information about congestion can be used, TCP has great opportunity to achieve better performance. For instance, queuing delay is a multi-bit indicator of congestion and it can quantitatively describe congestion. TCP Vegas and FAST are in the category of queuing delay based TCP. TCP Vegas adapts its window size $w(k)$ according

to [24]

$$w(k+1) = w(k) + \frac{1}{RTT} sgn(\alpha - x(k)q(k)), \tag{1.1}$$

where $sgn(x) = 1$ if $x < 0$, 0 if $x = 0$, and 1 if $x > 0$; and $q(k)$ is the queuing delay. In other words, TCP Vegas tries to maintain α packets queued in the network buffers. FAST is a high speed version of Vegas and the window size is adapted according to [24]:

$$w(k+1) = \gamma \left(\frac{dw(k)}{d + q(k)} + \alpha(w(k)q(k) \right) + (1-\gamma)w(k), \tag{1.2}$$

where d is the propagation delay and γ is a predefined parameter. It has been shown that FAST can stabilize the queue length very fast and utilize the bandwidth efficiently in high speed and long delay networks [24].

1.3.2 AQM

The basic ideas behind AQM is to control the traffic rate and buffer occupancy by schematically dropping packets before buffer overflows. There are two indicators that can be used to compute the dropping probability: queue length and aggregate traffic rate. Random Early Detection (RED) [13] is a typical queue length based AQM. Adaptive Virtual Queue (AVQ) [16] is a virtual queue based scheme. Random Exponential Marking (REM) [17], Proportional-Integral PI control [15] and State Feedback Control (SFQ) [18] use both queue length and aggregate traffic rate to compute the probability. The remainder of this section will provide a brief introduction to these AQM algorithms.

In RED, upon the arrival of the $k+1^{th}$ packet, the exponential averaged queue length is updated as

$$q_{ave}(k+1) = (1-w) \times q_{ave}(k) + w \times q(k), \tag{1.3}$$

where q is the instantaneous queue length and w is the averaging weight. When q_{ave} is less than the threshold min_{th}, no packets are dropped (or marked). When q_{ave} is between min_{th}

and max_{th} ($min_{th} < min_{th}$), the probability is computed as:

$$p = max_p \times \frac{q_{ave} - min_{th}}{max_{th} - min_{th}}. \tag{1.4}$$

When $q_{ave} > max_{th}$, all packets are dropped (or marked).

Due to the probabilistic dropping in RED, the TCP synchronization is greatly reduced. As a result, the link utilization is improved. RED can also make the queue length oscillate within a smaller range than that of DropTail. However, RED also has some disadvantages.

1. The average queue length in RED is proportional to the square of the number of TCP flows traversing the link. As the number of TCP flows increases, the queuing delay increases too. Stabilized-RED (SRED) [26] and Adaptive-RED (ARED) [25] were proposed to make the average queue length less dependent on the number of flows and maintain the average queue length within a smaller range than that of RED.

2. RED still has difficulty in stabilizing the queue length. As the round trip time and link capacity increase, RED will eventually operate in the instability region. Using control theory, Low *et al.* [28] demonstrated that inevitable instability is the result of the scheme itself. Ranjan *et al.* [27] showed that RED can exhibit chaotic behavior when the RED parameters fall into certain region. Hollot *et al.* [15] linearized the RED-TCP system and provided the guideline to set RED parameters so that it is locally stable when the number of TCP flows is large enough and the RTT is small enough.

REM [17] is another attractive AQM scheme in terms of achieving high link utilization, stable queue length and low packet loss. REM distinguishes itself from other AQM schemes by introducing the concept of "price". The "price" is the measurement of congestion at each link. Unlike RED, the "price" is decoupled from performance measures such as loss, queue length, and delay. At each link, REM continuously updates the value

of price and marks packets with exponential probability. At each link l, the price $p_l(t)$ is updated according to

$$p_l(t+1) = [p_l(t) - \gamma(\alpha_l q_l(t) + y_l(t) - C_l)]^+, \tag{1.5}$$

where $[z]^+ \triangleq \max(0, z)$, $y_l(t)$ is the total traffic arrival rate, and γ, α_l, and β_l are positive constants. REM can achieve stabler queue length than RED and sustain high link utilization. The local stable condition of REM was first studied in [29]. Chapter 3 of this dissertation will also provide further study of the stable condition for REM.

In AVQ [16], each router maintains a virtual queue with capacity $\tilde{C}$ less than the real link capacity C and the maximum length equal to that of the real queue [16]. Upon arrival of a new packet, the virtual queue capacity and the virtual queue length VQ are updated according to

$$\dot{\tilde{C}} = \alpha(\gamma C - y), \tag{1.6}$$

$$\dot{VQ} = \tilde{C} - y, \tag{1.7}$$

where y is the total arrival rate and γ is a positive number that is slightly less than one (e.g., 0.98). At arrival of each packet, the packet is dropped if $VQ > B$, where B is the maximum queue length of the real queue. It can be seen that AQV does not drop based on computed probability. Instead, it drops packets when the virtual queue overflows. AVQ controls the real queue length to zero at the cost of a smaller link utilization $\gamma < 1$, which is slightly less than one.

The PI controller [15] tries to maintain the queue length at a predefined level q_0, and updates the dropping probability according to

$$p(kT) = a\delta q(kT) - b\delta q((k-1)T) + p((k-1)T), \tag{1.8}$$

where a and b ($< a$) are two positive numbers, T is the sampling time, and $\delta q(kT) = q(kT) - q_0$. It can be seen that the PI controller updates the dropping probability after every T instead of doing so after arrival of every packet, and this greatly reduces the computation complexity. It has been shown that the PI controller stabilizes the queue length much better than RED and responds faster to traffics dynamics than RED.

1.4 Dissertation Organization

The remaining of this dissertation is organized as follows:

Chapter 2 presents the YSF aggregate packet scheduling scheme for EF PHB, derives its end-to-end delay bound, and compares its performance with that of the FIFO scheme. YSF eliminates the end-to-end delay divergence problem and provides a much smaller end-to-end delay bound than that of the FIFO scheduling scheme. YSF does not need to maintain per flow information and its scheduling complexity is independent of the number of flows traversing the link. Therefore, YSF is efficient, effective and scalable, and is a very promising candidate to schedule EF traffic.

Chapter 3 studies the local stability condition of REM. Local stability condition of REM is presented in the multi-link and multi-source network with arbitrary uniform feedback delay under the continuous time model. Various results provide valuable insight into how to set REM parameters so that a locally stable network can be achieved.

Chapter 4 presents VQR, a new AQM scheme, and provides its local stability condition in a network with a heterogeneous propagation delay for each TCP flow.

Chapter 5 presents the edge-based AQM (EAQM) scheme, and compares it with the traditional AQM schemes. Conventional AQM needs to be deployed over the entire network, and this requires tremendous upgrading to current Internet. EAQM is a pure edge based approach, in which only the edge routers are changed. It is easier and more economical to deploy EAQM than traditional AQM schemes. EAQM achieves similar or better performance in terms of queue length stability and responsiveness to traffic

dynamics. Moreover, EAQM can significantly reduce the throughput unfairness against TCP connections with longer RTTs.

Chapter 6 presents the Network-assist Packet Marking (NPM) scheme to fairly allocate bandwidth among TCP aggregates in the context of AF services. NPM can assure fair bandwidth allocation in both single and multiple bottleneck networks, regardless of the RTT, subscription rate, and the number flows of each aggregate.

Chapter 7 studies the throughput of High Speed TCP (TCP) in Optical Burst Switching (OBS) networks.

Chapter 8 concludes this dissertation and presents future works.

CHAPTER 2

YOUNGEST SERVE FIRST (YSF) AGGREGATED PACKET SCHEDULING FOR EF PHB

The EF PHB aims to guarantee bandwidth in both large and small time scale, but whether it can guarantee end-to-end QoS still remains unclear. It was believed that the end-to-end QoS in the network could be guaranteed if the link utilization is kept small enough (i.e., less than 50%). Recent studies [10] [11] show that the worst-case end-to-end delay bound for EF traffics through the network is proportional to $\frac{1}{1-(H-1)\alpha}$ if the FIFO scheme is applied, where H is the number of hops along the longest path of all the flows in the network, and α, the so called link utilization, is the ratio between the total amount of EF traffics on the link and the capacity of the corresponding link. It is clear that the worst-case delay is bounded only when $\alpha < \frac{1}{H-1}$. Thus, the provisioning power of traffic aggregation is significantly weakened. The reason behind the difficulty of obtaining bounded end-to-end delay for an arbitrary network topology is that in aggregated scheduling, packet delay not only depends on the traffic behavior of the flows sharing the same queue, but also on the traffic patterns in the whole network, even those occurred long time ago [10].

In this chapter, a new simple aggregated packet scheduling algorithm: Youngest Serve First (YSF) is proposed for EF traffics. The main objective of YSF is to achieve a better end-to-end delay bound for EF traffics than the FIFO aggregate scheduling scheme. In YSF, EF traffics are shaped at the network edge. A label value is used to indicate the packet state information and encode it in a certain field in each packet header. As packets travel through the network, the encoded information is updated. All the packets are scheduled based on the information carried in the header. This new approach not only has low computational complexity, but it also needs very limited number of bits ($\log_2 H$) to

carry the label in the packet headers. Most importantly, it can provide bounded end-to-end delay for any $\alpha < 1$ and any H.

The remainder of this chapter is organized as follows. Section 2.1 presents the introduction to the network model and terminology, and reviews the related background. In Section 2.2, the YSF scheme is presented and its end-to-end delay bound is derived based on Network Calculus Theory [12]. Finally, the summary is given in Section 2.3.

2.1 Network Model, Terminology, and Background

Assume that there are at least two classes of end-to-end flows [10] including the class of EF traffics, which are served with a strict priority over other classes of traffics. In general, EF services can be realized by the Guaranteed Rate (GR) scheme instead of being limited to priority queuing [11]. The proposed YSF can be extended to the GR framework. All network nodes are assumed to perform the same packet-scheduling algorithm based on the limited information carried in the packet headers. Before entering the network, EF traffic flow k is shaped at the network edge to conform to a token bucket with parameters (r^k, β^k), which is the traffic arrival curve satisfying $A^k(t_0, t_0+t) \leq r^k t + \beta^k$, where $A^k(t_0, t_0+t)$ is the total traffic from flow k released to the network during time interval $[t_0, t_0+t]$. Denote $F(I)$, as the set of flows traversing node I. It is assumed that for every $F(I)$ and I, the following conditions hold [10, 11]:

$$\sum_{k \in F(I)} r^k \leq \alpha C_I, \tag{2.1}$$

and

$$\sum_{k \in F(I)} \beta^k \leq \beta C_I, \tag{2.2}$$

where $\alpha(< 1)$ is the link utilization factor, β is the constraint on the burstiness of all flows through I, and C_I is the outgoing link capacity of I. According to [10], β is linearly dependent on α, and so let $\beta = \tau_0 \alpha$, where τ_0 is a constant. In this chapter, the fluid

traffic model is adopted, but this work can be easily extended to the packet traffic model. The effect of propagation delay is also assumed negligible. The following notations are adopted: d_i represents the maximum delay (worst-case) experienced by packet p at the i^{th} node along its path from the source to the destination, and D_i represents the maximum total delay (worst-case) experienced by packet p from the first node to the i^{th} node (inclusive) along the path, i.e., $D_i = \sum_{j=1}^{i} d_j$.

For simplicity, $A(t)$ is used to replace $A(t_0, t_0 + t)$ as the total traffic arrival curve. Denote $S(t)$ as the traffic service curve, which, in this case, is $S(t) = C(t)$. The amount of packets stored at each node is at most B, which is given by

$$B = A \bigoplus S(0), \tag{2.3}$$

where $\bigoplus$, the deconvolution, is defined by

$$A \bigoplus S(t) := \sup_{\tau \in R} A(t + \tau) - S(\tau). \tag{2.4}$$

If the total traffic arrive curve is

$$A(t) = \alpha C t + \beta C, \tag{2.5}$$

it can be verified that

$$B = \beta C. \tag{2.6}$$

In Figure 2.1, B is the maximum amount of packets stored, which is βC as mentioned above. t_B, the maximum queuing delay, is β in this case if the FIFO scheduling is adopted. t_d is the maximum burst length or the longest time for the system to clear the queue, as long as work conserving scheduling algorithms are adopted. t_d can be obtained by solving the following equation:

$$S(t_d) = A(t_d). \tag{2.7}$$

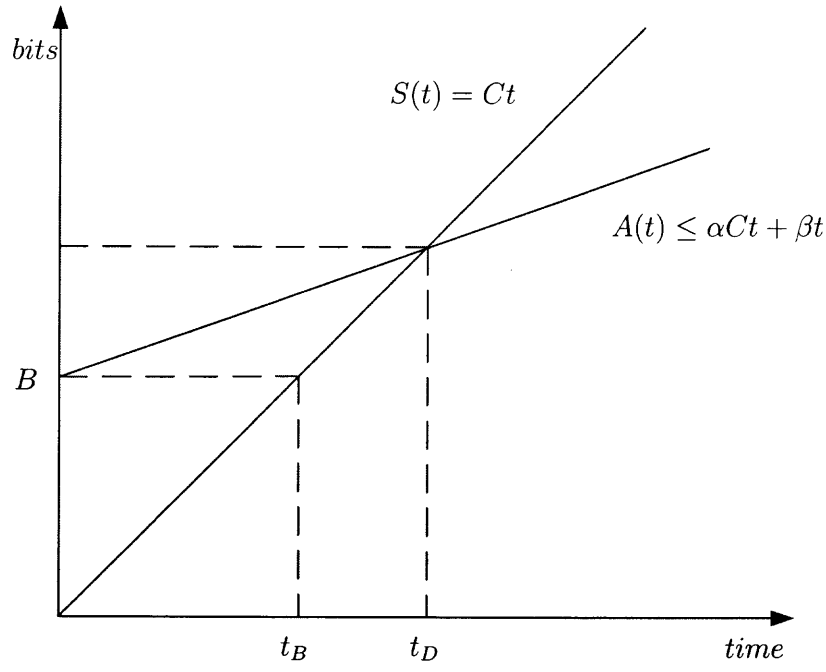

Figure 2.1: Illustration of network calculus.

2.2 The Youngest Serve First (YSF) Algorithm

Assume the edge node is the first hop for all EF traffics. The YSF algorithm works as follows: before entering the network, each packet is labeled with the number one, 1; at each node, the packet with the smallest label value is served first, and packets with the same label value are processed in the FIFO manner; the label on each packet is increased by one right before they are transmitted. The label value of a packet indicates the time it has spent in the network, or more precisely, the number of hops it has traversed through the network. That is why it is called Youngest Serve First (YSF) [1]

Naturally, the worst-case delay bound is experienced by packets with H hops from their source to the destination; H, as defined earlier, is the number of hops along the longest path of all the flows in the network. Consider a packet p. It can be seen that at different nodes along the path, p experiences different delay bound. At the first hop, p has the highest priority due to the smallest label value carried in its header. After p is transmitted from the

[1] Youngest Serve First is with respect to the hop count.

first hop node, the label value is increased to two at the second hop node. Thus, p cannot receive service as long as there are packets with label value one in that node. Intuitively, the longer the packet stays in the network, the larger the maximum delay it will experience at each node. In other words, $d_i \leq d_j$, for $1 \leq i \leq j \leq H$. In the rest of this chapter, traffics are grouped based on their labels carried in packet headers.

Definition 2.1 *Denote $F_i(I)$ as the set of flows which assume their j^{th} hop at node I. Therefore, $F(I) = F_1(I) \cup F_2(I) \cup ... \cup F_H(I)$. Define ρ_I^j and σ_I^j as*

$$\rho_I^j = \sum_{i \in F_j(I)} r^i, \tag{2.8}$$

$$\sigma_I^j = \sum_{i \in F_j(I)} \beta^i, \tag{2.9}$$

In other words, ρ_I^j is the sum of rates of flows that assume their j^{th} hop at node I, and σ_I^j is the sum of burstiness of flows that assume their j^{th} hop at node I.

Thus, Eq. 2.1 and 2.2 can be rewritten as:

$$\sum_{k \in F(I)} r^k = \sum_{j=1}^{H} \sum_{k \in F_j(I)} r^k = \sum_{j=1}^{H} \rho_I^j \leq \alpha C_I, \tag{2.10}$$

$$\sum_{k \in F(I)} \beta^k = \sum_{j=1}^{H} \sum_{k \in F_j(I)} \beta^k = \sum_{j=1}^{H} \sigma_I^j \leq \beta C_I. \tag{2.11}$$

Lemma 2.1 *The maximum delay experienced by packet p at its first hop in the network is $d_1 = \beta$ (is also D_1).*

***Proof*:** Suppose the first hop node is I. Since packet p has the smallest label value one, it has the highest priority in the system. Any other packets with a larger label value will not affect the service time of p. As a result, p experiences the worst-case delay when I is the first hop for all the flows traversing it. In this case, all packets have the same label value or

priority. Thus, the scheduling algorithm is just FIFO. According to Eqs. 2.10 and 2.11, the maximum overall traffic arrival curve at I is $A(t) = \alpha C_I(t) + \beta C_I$. According to Figure 2.1, based on Network Calculus Theory introduced before and the fluid traffic model used throughout this section, the delay bound is β. ■

Lemma 2.2 *The maximum delay experienced by packet p from its first hop node to the second hop node (inclusive) along the path is bounded by* $D_2 = \beta + \frac{\beta}{1-\alpha}$.

***Proof*:** Suppose the second hop node for p is I. According to the analysis in Lemma 2.1, only packets with label value of either one or two can affect the delay time of p at I. In order to obtain the worst-case delay bound for flow j at I, assume that only packets with label value one and two traverse node I. In other words, only packets experiencing their either first or second hop at I are considered. Since packets labeled with two may have already experienced delay D_1, the traffic arrival curve for those flows is $\rho_I^2(t + D_1) + \sigma_I^2$ instead of $\rho_I^2 t + \sigma_I^2$ [10]. For flows with the first hop at I, the arrival traffic curve is still $\rho_I^1 t + \sigma_I^1$. Therefore, the total arrival traffic curve at I is:

$$A(t) = \rho_I^1 t + \sigma_I^1 + \rho_I^2(t + D_1) + \sigma_I^2. \tag{2.12}$$

In order to get the maximum delay, the equalities in Eqs. 2.10 and 2.11 are used, which are

$$\sum_{j=1}^{2} \rho_I^j = \alpha C_I, \tag{2.13}$$

$$\sum_{j=1}^{2} \sigma_I^j = \beta C_I. \tag{2.14}$$

From Eqs. 2.13 and 2.14, Eq. 2.12 can be rewritten as

$$A(t) = \alpha(1-x)C_I t + x\alpha C_I(t + D_1) + \beta C_I = \alpha(1-x)C_I t + x\alpha C_I t + x\alpha C_I D_1 + \beta C_I, \tag{2.15}$$

where $x = \frac{\rho_I^2}{\alpha C_I}$. It can be seen that the burst size of the overall traffic is changed from $\beta\rho$ to $x\alpha C_I D_1$. According to the introduction in this section, when packet p arrives at node I, the maximum buffer occupied at I is $x\alpha C_I D_1 + \beta C_I$. All the packets in the buffer before p arrives will be served before p, since their priorities are not lower than p; packets that are labeled with two and arrive after p will not affect its departure time, because they carry the same labels as p and will be served in the FIFO order. However, those packets labeled with one, of which the arrival rate is $(1-x)\alpha C_I$, will affect the departure time of p even they enter the queue after p. Thus, the effective total arrival traffic curve that can determine the departure time of p is

$$C_I d_2 = A_{eff}(d_2) = (1-x)\alpha C_I d_2 + (x\alpha C_I D_1 + \beta C_I). \tag{2.16}$$

Thus,

$$d_2 = \frac{x\alpha D_1 + \beta}{1-(1-x)\alpha} = \frac{(x\alpha+1)\beta}{1-(1-x)\alpha}. \tag{2.17}$$

Since

$$\frac{\partial d_2}{\partial x} = \frac{-\alpha^2}{\{1-(1-x)\alpha\}^2}\beta < 0, \tag{2.18}$$

when $x = 0$, d_2 reaches its maximum

$$d_2 = \frac{\beta}{1-\alpha}. \tag{2.19}$$

Hence,

$$D_2 = D_1 + d_2 = \beta + \frac{\beta}{1-\alpha}. \tag{2.20}$$

After packet p takes its second hop and moves to the j^{th} (> 2) node, there are possibly more packets with smaller label values or higher priorities at that node. Intuitively, p could experience longer delay at those nodes. Next, in Theorem 2.1, the delay bound as p moves closer to its destination is derived.

Theorem 2.1 *The maximum delay experienced by packet p from its first hop node to the k^{th} hop node (inclusive) along the path is bounded by $D_k = D_{k-1} + \frac{\beta+\alpha D_{k-2}}{1-\alpha}$, for any $k \geq 3$*

***Proof*:** Induction will be used to complete the proof. Define $D_0 = 0$. From Lemma 2.1 and Lemma 2.2,

$$D_2 = D_1 + d_2 = \beta + \frac{\beta}{1-\alpha} = D_1 + \frac{\beta + \alpha D_0}{1-\alpha}. \tag{2.21}$$

Assume

$$D_k = D_{k-1} + \frac{\beta + \alpha D_{k-2}}{1-\alpha}. \tag{2.22}$$

Next, it will show that the expression holds for $k+1$. Let node I be the $(k+1)^{th}$ hop of packet p, and thus only packets with label not larger than $k+1$ can affect the delay of p at I. In other words, only packets assuming their $j^{th}(j \leq k+1)$ hop at node I will be considered. The overall arrival traffic curve can be written as

$$A(t) = \sum_{j=1}^{k+1} \{\rho_I^j(t + D_{j-1}) + \sigma_I^j\}. \tag{2.23}$$

Define $x_j = \frac{\rho_I^j}{\alpha C_I}$, and also note that $\sum_{j=1}^{k+1} x_j = 1$. Then Eq. 2.23 is rewritten as

$$\begin{aligned}
A(t) &= \sum_{j=1}^{k+1} \{\rho_I^j(t + D_{j-1}) + \sigma_I^j\} = \sum_{j=1}^{k+1} \rho_I^j(t + D_{j-1}) + \sum_{j=1}^{k+1} \sigma_I^j \\
&= \sum_{j=1}^{k+1} x_j \alpha C_I(t + D_j) + \beta C_I \\
&\leq x_{k+1}\alpha C_I t + x_{k+1}\alpha C_I D_k + \sum_{j=1}^{k} x_j \alpha C_I t + \sum_{j=1}^{k} x_j \alpha C_I D_{k-1} + \beta C_I \\
&= x_{k+1}\alpha C_I t + x_{k+1}\alpha C_I D_k + (1 - x_{k+1})\alpha C_I t + \sum_{j=1}^{k} x_j \alpha C_I D_{k-1} + \beta C_I \\
&= x_{k+1}\alpha C_I t + (1 - x_{k+1})\alpha C_I t + [x_{k+1}\alpha C_I D_k + (1 - x_{k+1})\alpha C_I D_{k-1} + \beta C_I]
\end{aligned} \tag{2.24}$$

The inequality in Eq. 2.24 is based on the fact that $D_i < D_{k-1}$ if $i < k-1$. The third term in the last equality stands for the maximum traffics queued in the system when packet p joins the queue. The first term can be viewed as the traffics which assume the $(k+1)^{th}$ hop at node I, and arrive after p, and thus cannot affect p's departure time. The second term represents the traffics with smaller label values, which also arrive after p, that they can affect the departure time of p. Using the similar argument in the proof of Lemma 2.2, the effective arrival traffic curve can be rewritten as

$$A_{eff}(t) = (1-x_{k+1})\alpha C_I t + x_{k+1}\alpha C_I D_k + (1-x_{k+1})\alpha C_I D_{k-1} + \beta C_I. \tag{2.25}$$

From the following equation

$$C_I d_{k+1} = A_{eff}(d_{k+1}) = (1-x_{k+1})\alpha C_I d_{k+1} + x_{k+1}\alpha C_I D_k + (1-x_{k+1})\alpha C_I D_{k-1} + \beta C_I \tag{2.26}$$

d_{k+1} can be expressed as

$$d_{k+1} = \frac{x_{k+1}\alpha D_k + (1-x_{k+1})\alpha D_{k-1} + \beta}{1-(1-x_{k+1})\alpha}. \tag{2.27}$$

Thus,

$$\begin{aligned}\frac{\partial d_{k+1}}{\partial x_{k+1}} &= \frac{\alpha[(1-\alpha)D_k - D_{k-1} - \beta]}{[1-(1-x_{k+1})\alpha]^2} \\ &= \frac{\alpha[(1-\alpha)(D_{k-1} + \frac{\alpha D_{k-2}+\beta}{1-\alpha}) - D_{k-1} - \beta]}{[1-(1-x_{k+1})\alpha]^2} \\ &= \frac{\alpha(D_{k-2} - D_{k-1})}{[1-(1-x_{k+1})\alpha]^2} < 0.\end{aligned} \tag{2.28}$$

Eq. 2.22 has been used to reach the third equality in Eq. 2.28. So, when $x_{k+1} = 0$, d_{k+1} reaches its maximum value. From Eq. 2.27,

$$d_{k+1} = \frac{\alpha D_{k-1} + \beta}{1-\alpha} \tag{2.29}$$

and

$$D_{k+1} = D_k + d_{k+1} = D_k + \frac{\alpha D_{k-1} + \beta}{1-\alpha} \tag{2.30}$$

From the recursive relationship in Eq.2.30 with the initial conditions stated in Lemma 2.1 and Lemma 2.2, it can be shown

$$D_j = \frac{\beta}{\alpha}[\frac{r_2 - \alpha - 1}{r_2 - r_1} r_1^j + \frac{r_1 - \alpha - 1}{r_1 - r_1} r_1^j - 1] \qquad j \geq 3 \tag{2.31}$$

where $r_{1,2} = \frac{1-\alpha \pm \sqrt{-3\alpha^2+2\alpha+1}}{2(1-\alpha)}$ are the roots of the following quadratic equation

$$r^2 - r - \frac{\alpha}{1-\alpha} = 0. \tag{2.32}$$

It can be shown from Eq.2.31 with further algebraic manipulation that $D_H \sim H\beta$ when α is very small, and $D_H \sim (1-\alpha)^{H/2}$ when $\alpha \approx 1$. Based on the above analysis, it can concluded that the link utilization, which is independent of H, can approach one, and the end-to-end delay is also bounded at the same time. The end-to-end delay bound for FIFO [10] is $\frac{H\beta}{1-\alpha(H-1)}$, if the fluid traffic model is applied. Note that β is denoted by τ in [10]. Figure 2.2 shows the performance comparison between YSF and FIFO with $H = 10$. The vertical axis is the number of time units (in terms of τ_0). With a given link utilization α, YSF performs much better than FIFO, especially when α is large.

It may seem very natural to adopt another scheme, namely Oldest Serve First (OSF), in which the packets with the largest label value are served first instead of being served last in YSF. Next, it will show that YSF performs better than OSF.

Lemma 2.3 *In OSF, for any packet p with H hops to its destination, the worst-case delay experienced till its* $(H-1)^{th}$ *hop (inclusive) is* $D_{H-1} = \frac{(H-1)\beta}{1-\alpha H}$.

***Proof*:** In order to derive the worst-case delay, assume that the node I is the $k^{th}(1 \leq k \leq H-1)$ hop node along p's path to its destination, and all packets from other flows assume their H^{th} hop at I. According to OSF, packet p has the lowest priority and can only be served when no other packets are in the corresponding node. Since other flows may all experience the maximum delay D_{H-1}, the total arrival traffic curve is

$$A(t) = \alpha C_I(t + D_{H-1}) + \beta C_I. \tag{2.33}$$

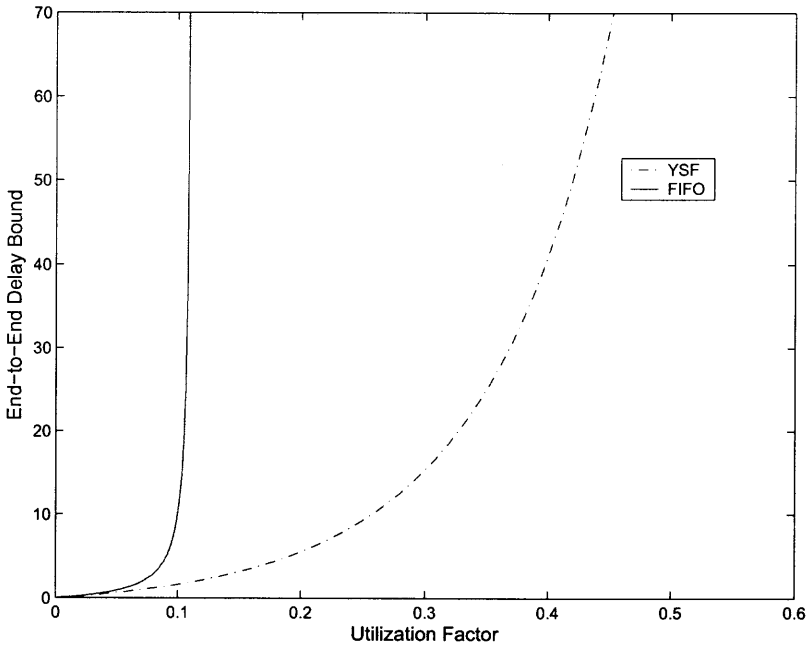

Figure 2.2: Performance comparison between FIFO and YSF ($H = 10$).

Then, d_k, the delay bound of p experienced at the k^{th} node can be obtained by solving the following equation:

$$S(d_k) = C_I d_k = A(d_k) = \alpha C_I(d_k + D_{H-1}) + \beta C_I. \tag{2.34}$$

Thus,

$$d_k = \frac{\alpha D_{H-1} + \beta}{1 - \alpha}. \tag{2.35}$$

On the other hand,

$$D_{H-1} = \sum_{k=1}^{H-1} d_k = (H-1)\frac{\alpha D_{H-1} + \beta}{1 - \alpha}. \tag{2.36}$$

Hence,

$$D_{H-1} = \frac{(H-1)\beta}{1 - \alpha H} \tag{2.37}$$

From Lemma 2.3, it can be seen that D_{H-1} is bounded only when $\alpha < \frac{1}{H}$. Since α can approach one in YSF, YSF achieves higher link utilization. From Eq. 2.37, it can also be seen that in OSF, when approaches one, the end-to-end delay bound goes to infinity. However, in YSF, the end-to-end delay bound always remains finite as long as $\alpha < 1$. Thus, YSF also performs better than OSF in terms of the end-to-end delay bound.

2.3 Summary

In this chapter, a new aggregated traffic scheduling scheme, Youngest Serve First (YSF) is proposed, and its end-to-end delay bound is derived. YSF has been proven to have the following merits. First, the link utilization α in YSF can approach one, regardless of the network topology and the value of H; second, the end-to-end delay bound in YSF is much smaller than that in FIFO, and thus better end-to-end delay bound can be guaranteed. Even with an additional required complexity, which is rather low, the above merits warrant YSF preferable over FIFO. At each node, there are H different label values. Thus, at most H queues are required, which correspond to different label values. Packets are placed into different queues based on their label values and the backlogged queue with the smallest label value is served first. Therefore, YSF is scalable because only H queues are needed, no matter how many flows traverse each node. Note that only $\log_2 H$ bits are required to encode the label. Either the TTL field or the TOS field in the IP header can be used to realize YSF, but this issue is beyond the scope of this chapter. Ideas from timestamp based scheduling algorithm such as WFQ and WF^2Q may be incorporated in designing aggregated traffic scheduling schemes. However, YSF possesses the following merit not shared by the timestamp-based approaches: all the nodes in the network need not be synchronized in time, and timestamps need not be computed and updated.

YSF can be incorporated in Diffserv, which is the emerging service architecture for the Internet. YSF can be an alternative to the FIFO scheme, which is currently employed in EF traffic scheduling. EF is usually assumed to support delay-sensitive applications, e.g.,

audio streaming. As addressed in [10, 11], the FIFO scheme can provide a strict end-to-end delay bound only for a small link utilization and limited hop count. This shortcoming of FIFO can severely limit the QoS provisioning required by many delay sensitive applications using EF. Retaining the simplicity of aggregate scheduling, YSF provides strict and low end-to-end delay bound for any hop count and link utilization (less than one).

CHAPTER 3

THE LOCAL STABLE CONDITION FOR RANDOM EXPONENTIAL MARKING (REM)

TCP is one of the major transport protocols over the current Internet. TCP provides end-to-end congestion control by dynamically adjusting the transmitting rate based on the congestion feedback. In the network, packets are either dropped or marked when congestion happens. Before RED (Random Early Detection) [13] was proposed, the major congestion control scheme is DropTail, which suffers from synchronization of senders and oscillation of buffer occupation. RED tries to overcome the shortcomings of DropTail by dropping packets in a probabilistic manner before the buffer overflows. ECN (Explicit Congestion Notification) [14] notifies users of congestion by marking packets probabilistically instead of dropping them. The users react to the marked packets as if packet loss is detected.

In order to achieve desired properties such as maintaining fairness among users and stabilizing queue length in the network, packet marking or dropping schemes should be designed very carefully. Adaptive Queue Management (AQM) schemes [16, 15, 17] have been proposed to mark or drop packets intelligently. One of the major challenges faced by AQM is how to achieve a stable system. Recent works [27, 28] have discussed the dynamic behavior of TCP-AQM in the framework of a feedback control system. It has been shown [27] that TCP-RED can exhibit chaotic behavior when RED parameters fall into a certain region. It was also demonstrated in [28] that instability of TCP-RED is the inevitable result of the scheme itself.

Random Exponential Marking (REM) [17] was a very attractive AQM scheme in terms of achieving high link utilization, stable queue length and low packet loss. REM distinguishes itself from other AQM schemes by introducing the concept of *price*. The

price is the measurement of congestion at each link. Unlike RED, the *price* is decoupled from performance measures such as loss, queue length or delay. At each link, REM continuously updates the value of price (the update scheme is explained later in Section 3.2) and marks packets with exponential probability. Readers are referred to [17] for more details.

Like other AQM algorithms, REM has to address the stability problem. Extensive simulations have shown that REM exhibits highly stable behavior in a wide range of network configurations. In [17], local stable condition was studied under the discrete time model without considering feedback delay. In [29] and [30], global stability was proven for zero feedback delay in continuous and discrete time model, respectively. In [31], for the first time, local stability of REM with feedback delay was investigated. Nevertheless, [31] used the discrete time model and only presented the analytical result of one- and two- step uniform feedback delay.

The major contribution of this chapter is the derivation of the local stable conditions of REM with any value of uniform feedback delay in a multi-link and multi-source network by using the continuous time model.

The remainder of this chapter is organized as follows: Section 3.1 provides the necessary background and problem formulation, Section 3.2 presents the results for REM local stable conditions, and summary is presented in Section 3.3.

3.1 Background and Problem Formulation

Consider a network with L links, each with capacity $C_i(i = 1, 2, ...L)$. Assume there exist S TCP sources, which share the L links. The routing policy is expressed by an $S \times L$ matrix **A** with elements a_{ij} defined as:

$$a_{ij} = \begin{cases} 1 & \text{if source } j \text{ uses link } i \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

The following notations are adopted for the rest of this chapter:

$p_i(t)$: The non-negative price for link i at time t. The corresponding vector form is

$\mathbf{p}(t) = (p_1(t), p_2(t), ...p_L(t))$.

$q_i(t)$: The queue length for link i at time t. The corresponding vector form is

$\mathbf{q}(t) = (p_1(t), p_2(t), ...p_L(t))$.

$x_j(t)$: The rate of source j at time $t(j = 1, 2, ..., S)$.

$r_i(t)$: The rate of all sources crossing link i at time t.

$y_j(t)$: The total price of all links used by source j at time t.

For the sake of brevity, the variable t in the above notations may be omitted in subsequent discussions. From the above notations,

$$r_i = \sum_{j=1}^{S} a_{ij} x_j \tag{3.2}$$

and

$$y_i = \sum_{i=1}^{L} a_{ij} p_i. \tag{3.3}$$

According to [29], TCP source j adjusts its sending rate by maximizing $U_j(x_j) - p_j x_j$; here, $U_j(x_j)$ is the utility function of source j and is strictly concave increasing. The utility function of TCP Reno [28] is

$$U_j(x_j) = \frac{\sqrt{2}}{d_j} \arctan(\frac{d_j x_j}{\sqrt{2}}), \tag{3.4}$$

and for TCP Vegas [28]

$$U_j(x_j) = \log(x_j). \tag{3.5}$$

In both versions of TCP, source j can be modeled to adjust its transmission rate in a smoothed manner of the following adjustment [17, 32, 33]

$$x_j(t) = [U_j']^{-1}(y_j(t - d_j)) = [U_j']^{-1}(\sum_{i=1}^{L} a_{ij} p_i(t - d_j)), \tag{3.6}$$

where $[U_j']^{-1}$ is the inverse function of the derivative of utility function U_j (it exists since U_j is strictly concave increasing). As in [17], assume the forward delay is zero and the backward delay is d_j. To simplify the analysis, only the case of homogeneous delay is considered, implying that $d_j = d$ for all j. Thus, Eq. 3.6 is reduced to

$$x_j(t) = [U_j']^{-1}(\sum_{i=1}^{L} a_{ij} p_i(t-d)). \tag{3.7}$$

The continuous time model [29] is adopted to describe the dynamics of REM:

$$\frac{dq_i(t)}{dt} = \begin{cases} r_i - C_i & \text{if } q_i(t) > 0 \\ [r_i - C_i]^+ & \text{if } q_i(t) = 0 \end{cases} \tag{3.8}$$

$$\frac{dp_i(t)}{dt} = \begin{cases} \gamma(\alpha q_i + r_i - C_i) & \text{if } p_i(t) > 0 \\ \gamma[\alpha q_i + r_i - C_i]^+ & \text{if } p_i(t) = 0 \end{cases} \tag{3.9}$$

where $[z]^+ \triangleq \max(0, z)$, and are positive constants. Eq. 3.7, 3.8 and 3.9 can be interpreted as a gradient projection algorithm used for solving an optimization problem. There is tradeoff between the selection of large α and small α : a small α leads to a high link utilization at the cost of a large transient queue length; on the other hand, a large α allows a small transient queue length and a low link utilization. γ corresponds to the step size of the gradient algorithm. Readers are referred to [17] for detailed discussion on γ and α. It is difficult to analyze the above system due to the non-linear term $[\cdot]^+$.

3.2 Cost Model and Performance Evaluation

Eq. 3.8 and 3.9 define a nonlinear dynamic system. Assume $rank(\mathbf{A}) = L$. Let $(\mathbf{p}^*, \mathbf{q}^*)$ be the equilibrium of the original system. It can be shown [17] that $x_j^* = [U_j']^{-1}(y_j^*) = [U_j']^{-1}(\sum_{i=1}^{L} a_{ij} p_i^*)$, $r_i^* = C_i$ and $q_i^* = 0$ at the bottleneck link. In order to simplify the analysis, a linearized version of the original system is developed. With the above properties

of the equilibrium, Eq. 3.8 and 3.9 can be rewritten as:

$$\frac{d(q_l(t) - q_l^*)}{dt} = r_l(t) - C_l, \tag{3.10}$$

$$\frac{d(p_l(t) - p_l^*)}{dt} = \gamma[\alpha(q_l(t) - q_l^*) + r_l(t) - r_l^*]. \tag{3.11}$$

Here, only the linear terms in the above equations are kept. If the similar procedure in [17] is followed, the first order Taylor expansion around (p^*, q^*) can be used to further simplify the system. $[U_j^{'}]^{-1}(y_j)$ can be expressed as:

$$\begin{aligned}[U_j^{'}]^{-1}(y_j) &= [U_j^{'}]^{-1}(y_j^*) + [[U_j^{'}]^{-1}]^{'}(y_j^*)(y_j - y_j^*) \\ &= [U_j^{'}]^{-1}(y_j^*) + \frac{1}{U_j^{''}(x_j^*)}(y_j - y_j^*).\end{aligned} \tag{3.12}$$

Thus,

$$\begin{aligned}r_i(t) &= \sum_{j=1}^{S} a_{ij}x_j(t) = \sum_{j=1}^{S} a_{ij}[U_j^{'}]^{-1}y_j(t-d) \\ &= \sum_{j=1}^{S} a_{ij}[U_j^{'}]^{-1}(y_j^*) + \sum_{j=1}^{S} a_{ij}\frac{1}{U_j^{''}(x_j^*)}(y_i(t-d) - y_i^*).\end{aligned} \tag{3.13}$$

According to the property $r_i^* = C_i$ at the bottleneck link,

$$C_i = \sum_{j=1}^{S} a_{ij}x_j^* = \sum_{j=1}^{S} a_{ij}[U_j^{'}]^{-1}(y_j^*). \tag{3.14}$$

Combining Eqs. 3.13 and 3.14,

$$r_i(t) - C_i = \sum_{i=1}^{S} a_{ij}\frac{1}{U_j^{''}(x_j^*)}(y_j(t-d) - y_j^*). \tag{3.15}$$

According to Eq. 3.3

$$y_j(t-d) - y_j^* = \sum_{l=1}^{L} a_{lj}(p_l(t-d) - p_l^*). \tag{3.16}$$

Combining Eqs. 3.15 and 3.16,

$$r_i(t) - C_i = \sum_{l=1}^{L} \sum_{j=1}^{S} a_{ij} a_{lj} \frac{1}{U_j''(x_j^*)} (p_l(t-d) - p_l^*). \tag{3.17}$$

Denote $\eta_j = \frac{-1}{U_j''(x_j^*)}$ and define the diagonal matrix

$$\mathbf{K} = diag\{\eta_1, \eta_2 ..., \eta_S\}, \tag{3.18}$$

and the following new variables:

$$\bar{q}_i(t) = q_i(t) - q_i^*, \tag{3.19}$$

$$\bar{p}_i(t) = p_i(t) - p_i^*. \tag{3.20}$$

Using Eq. 3.16-3.20, Eq. 3.10 and 3.11 can be rewritten into the following matrix forms:

$$\frac{d\bar{\mathbf{q}}(t)}{dt} = -\mathbf{A}\mathbf{K}\mathbf{A}^T \bar{\mathbf{p}}(t-d), \tag{3.21}$$

$$\frac{d\bar{\mathbf{p}}(t)}{dt} = \gamma[\alpha \bar{\mathbf{q}}(t) - \mathbf{A}\mathbf{K}\mathbf{A}^T \bar{\mathbf{p}}(t-d)]. \tag{3.22}$$

Taking the Laplace-transforms of Eqs. 3.21 and 3.22,

$$s\bar{\mathbf{q}}(s) = -e^{-sd}\mathbf{A}\mathbf{K}\mathbf{A}^T \bar{\mathbf{p}}(s), \tag{3.23}$$

$$s\bar{\mathbf{p}}(s) = \gamma[\alpha \bar{\mathbf{q}}(s) - e^{-sd}\mathbf{A}\mathbf{K}\mathbf{A}^T \bar{p}(s)]. \tag{3.24}$$

Substituting Eq. 3.23 into Eq. 3.24,

$$[s\mathbf{I} + \gamma e^{-sd}(1 + \frac{\alpha}{s})\mathbf{A}\mathbf{K}\mathbf{A}^T]\bar{\mathbf{p}}(s) = 0, \tag{3.25}$$

or equivalently

$$[s\mathbf{I} + \gamma e^{-sd}(1 + \frac{\alpha}{s})\mathbf{A}\mathbf{K}\mathbf{A}^T]\bar{\mathbf{p}}(s) = 0. \tag{3.26}$$

In order to have a non-trivial solution of $\bar{p}(s)$, it is required that

$$[s\mathbf{I} + \gamma e^{-sd}(1 + \frac{\alpha}{s})\mathbf{AKA}^T]\bar{\mathbf{p}}(s) = 0, \tag{3.27}$$

which is called the characteristic equation. Since $\mathbf{A}$ is an $L \times S$ matrix with rank L, and $\mathbf{K}$ is an $S \times S$ diagonal matrix, $\mathbf{AKA}^T$ is an $L \times L$ positive definite matrix. Therefore, it can be concluded that $0 < \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_L$, where λ_1, λ_2,..., λ_L are the eigenvalues of $\mathbf{AKA}^T$. Let $\mathbf{\Lambda} = diag(\lambda_1, \lambda_2, ..., \lambda_L)$. Eq. 3.27 is equivalent to Theorem 7 in [34], i.e.,

$$\det[s\mathbf{I} + \gamma e^{-sd}(1 + \frac{\alpha}{s})\mathbf{\Lambda}] = 0, \tag{3.28}$$

or

$$s + \gamma e^{-sd}(1 + \frac{\alpha}{s})\lambda_k = 0 \qquad k = 1, 2...L. \tag{3.29}$$

In order to have a stable system, all of the L roots of Eq. 3.29 should be on the left-half plane. Next, stable conditions for the above system will be derived.

Theorem 3.1 *If the feedback delay d is zero, the system is always stable.*

***Proof*:** If d is zero, from Eq. 3.29,

$$s^2 + \gamma s + \gamma\alpha\lambda_k = 0. \tag{3.30}$$

Thus,

$$s = \frac{-\gamma \pm \sqrt{\gamma^2 - 4\gamma\alpha\lambda_k}}{2}. \tag{3.31}$$

It can be seen that s is always on the left-half plane regardless of the value of γ.

Theorem 3.2 *If the feedback delay d satisfies $d < D$, the system is stable. Here, $D = \frac{\frac{\pi}{2} - \arctan(\frac{\alpha}{\theta})}{\theta}$, $\theta = \sqrt{\frac{\gamma^2\lambda^2 + \sqrt{\gamma^4\lambda^4 + 4\gamma^2\alpha^2\lambda^2}}{2}}$, and $\lambda = \max_k\{\lambda_k\}$.*

***Proof*:** For a fixed value of k, let $\tilde{d}_k$ be the smallest number such that the root for Eq. 3.29 stays to the right on the imaginary axis. According to Theorem 3.1, when d is zero, the

roots of Eq. 3.29 are all on the left-half plane. Therefore, if $d < \min_k \tilde{d}_k$, then all the roots of Eq. 3.29 are on the left-half plane. The similar procedure of the proof of THEOREM 2 in [35] can be used to determine $\tilde{d}_k$. Let the root of Eq. 3.29 be $s_k = j\theta_k$; here, j is the unit of the imaginary number instead of the index used before. Since the roots on the imaginary axis are complementary, only $\theta_k > 0$ is considered. Hence, Eq. 3.29 becomes:

$$\frac{\gamma e^{-j\theta\tilde{d}_k}(1+\frac{\alpha}{j\theta})\lambda_k}{j\theta} = -1. \tag{3.32}$$

There are two conditions on the magnitude and angle, respectively:

$$\left|\frac{\gamma e^{-j\theta\tilde{d}_k}(1+\frac{\alpha}{j\theta_k})\lambda_k}{j\theta_k}\right| = 1 \tag{3.33}$$

and

$$\angle\frac{\gamma e^{-j\theta\tilde{d}_k}(1+\frac{\alpha}{j\theta_k})\lambda_k}{j\theta_k} = (2n+1)\pi, \qquad n = 0, \pm1, \pm2... \tag{3.34}$$

The condition on the magnitude in Eq. 3.33 leads to

$$\theta_k = \sqrt{\frac{\gamma\lambda_k\sqrt{\gamma^2\lambda_k^2+4\alpha^2}+\gamma^2\lambda_k^2}{2}} \tag{3.35}$$

From Eq. 3.34,

$$\theta_k\tilde{d}_k + \arctan(\frac{\alpha}{\theta_k}) + \frac{\pi}{2} = (2n+1)\pi, \qquad n = 1, 2, ... \tag{3.36}$$

Taking $n = 0$ in Eq. 3.36,

$$\tilde{d}_k = \frac{\frac{\pi}{2} - \arctan(\frac{\alpha}{\theta_k})}{\theta_k}. \tag{3.37}$$

It can be verified that d_k is an increasing function of λ_k. Therefore,

$$\min_k \tilde{d}_k = \min_k \left\{\frac{\frac{\pi}{2} - \arctan(\frac{\alpha}{\theta_k})}{\theta_k}\right\} = \frac{\frac{\pi}{2} - \arctan(\frac{\alpha}{\theta})}{\theta}, \tag{3.38}$$

where

$$\theta = \sqrt{\frac{\gamma\lambda\sqrt{\gamma^2\lambda^2+4\alpha^2}+\gamma^2\lambda^2}{2}}. \tag{3.39}$$

Finally,

$$D = \min_k \tilde{d}_k = \min_k \{\frac{\frac{\pi}{2} - \arctan(\frac{\alpha}{\theta_k})}{\theta_k}\}. \tag{3.40}$$

3.3 Discussions

According to Theorem 3.2, the system is stable if $d < D$. This condition can be rewritten as

$$\alpha d < \alpha D = \frac{\alpha}{\theta}\{\frac{\pi}{2} - \arctan(\frac{\alpha}{\theta})\} = \frac{1}{x}\{\frac{\pi}{2} - \arctan(\frac{1}{x})\}, \tag{3.41}$$

where

$$x = \sqrt{\frac{(\gamma\lambda/\alpha)^2 + \gamma\lambda/\alpha\sqrt{4 + (\gamma\lambda/\alpha)^2}}{2}}, . \tag{3.42}$$

Figure 3.1 shows the local stable regions for REM based on Eq. 3.40. The horizontal axis is $\gamma\lambda/\alpha$; the vertical axis is αd. If the system falls into the region below the curve, it is locally stable. Otherwise, it is not stable. It can be seen that the local stability of REM depends on four parameters: γ, α, d and λ. γ and α are parameters set by the REM algorithm; λ and d are parameters determined by the network topology and the routing policy, which may not be controlled by the REM algorithm. In other words, although some pairs of γ and α lead to local stability under some network conditions, they can cause instability under other network conditions. Therefore, the value of γ and α should be set very carefully.

3.4 Summary

In this chapter the continuous time model is used to investigate the local stable conditions for REM in the multi-link and multi-source network, in which all sources have the same feedback delay. The study shows that the local stability of REM depends on both the algorithm parameter settings and network conditions. Several aspects still require further investigation. First, the approach in this work is based on the linearization of a non-linear system like other research on AQM. Nevertheless, Reference [27] has demonstrated the

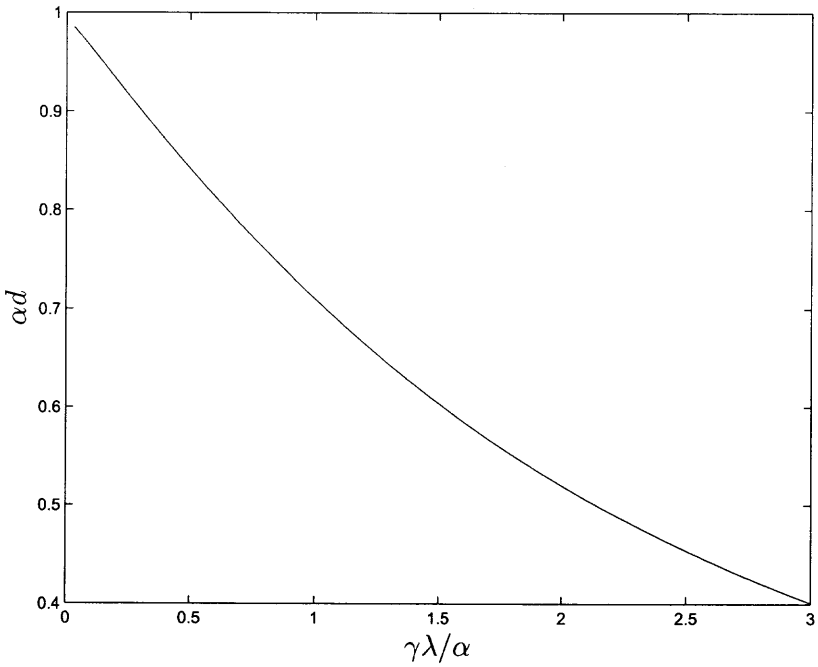

Figure 3.1: The local stable region for the parameters of REM.

important role of the non-linearity in AQM. Therefore, exploration of the non-linearity effect is very critical to the full understanding of the performance of REM and other AQM schemes. Second, the approach assumes the homogeneity of the feedback delay. In the real world, different users can experience different propagation delay and queuing delay. As a result, the feedback delay can exhibit significant heterogeneity. The investigation of this effect will be reported in the future research.

CHAPTER 4

VQR: VIRTUAL QUEUE AND RATE BASED AQM SCHEME

Active Queue Management (AQM) has been a very active research area in recent years. Many AQM mechanisms have been proposed, e.g., random early detection (RED) [13], random exponential marking (REM) [17], PI controller [15], adaptive virtual queue (AVQ) [16], and state feedback control (SFC) [18]. AQM schemes control the traffic rate and buffer occupancy by schematically dropping packets. If Explicit Congestion Control (ECN) is enabled, packets are marked instead of being dropped. Without loss of generality, ECN is assumed to be enabled in this chapter. End TCP users adapt their transmission rates based on the marking feedback from AQM.

RED is a queue length based AQM that marks packets with probability proportional to the current average queue length. AVQ is a typical rate based scheme. REM, PI, and SFC use the queue length and the aggregate flow rate to compute the marking probability.

The TCP/AQM system has been modeled as a close loop control system. One of the major concerns about such a system is its stability property. References [16, 15, 18] studied the local stability conditions for the network with PI, AVQ, and SFC, respectively. These works only considered a network of a single bottleneck link with homogeneous round trip times, and neglected the backward propagation delays. Reference [36] took the heterogeneous round trip time and backward propagation delay into consideration, and provided the local stability condition for RED in a network with a single bottleneck link.

In this chapter, a virtual queue and rate based AQM scheme (referred to as VQR) is proposed. VQR uses the virtual queue size and the aggregate flow rate to compute the marking probability. The local stability condition of the VQR scheme is provided for a network with an arbitrary topology rather than a single bottleneck link, with heterogeneous

rather than homogeneous round trip times, and with the consideration of rather than negligence of the backward propagation delays.

The remainder of this chapter is organized as follows. Section II presents the framework of the TCP/AQM dynamic model. Section III presents the proposed VQR scheme and the corresponding local stability condition. Section IV presents the conclusions.

4.1 TCP/AQM Dynamics

The work in [36] is adopted to present the TCP/AQM dynamic model. Consider a network with L links, each with capacity $C_l(\ l = 1, 2, ...L)$. Assume there exist S TCP sources, which share the L links. The routing policy is expressed by an $L \times S$ matrix $\mathbf{R}$ with elements R_{ij} defined as:

$$R_{il} = \begin{cases} 1 & \text{if source } i \text{ uses link } l \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

Each link l marks packets with probability $p_l(t)$ at time t. Each TCP source i is associated with a round trip time $\tau_i(t)$:

$$\tau_i(t) = d_i + \sum_l R_{li} \frac{b_l(t)}{C_l}, \tag{4.2}$$

where d_i is the round trip propagation delay for the source i and $b_l(t)$ is the queue length of link l at time t. The round trip time delay $\tau_i(t)$ can also be expressed as

$$\tau_i(t) = \tau_{li}^f(t) + \tau_{li}^b(t), \tag{4.3}$$

where $\tau_{li}^f(t)$ is the forward delay from source i to link l and $\tau_{li}^b(t)$ is the backward delay from link l back to the source i. The transmitting rate of the source is modeled as

$$x_i(t) = \frac{w_i(t)}{\tau_i(t)}, \tag{4.4}$$

where $w_i(t)$ is the window size of source i. The aggregate transmission rate at link l is

$$y_l(t) = \sum_i R_{li} x_i(t - \tau_{li}^f(t)). \tag{4.5}$$

With the assumption that $p_l(t)$ is small, the end-to-end marking probability $q_l(t)$ for source i can be obtained by summing up the marking probability at each link traversed by i

$$q_i(t) = \sum_l R_{li} p_l(t - \tau_{li}^b(t)). \tag{4.6}$$

The fluid TCP model studied in [15] and [36] is adopted:

$$\dot{w}_i(t) = \frac{x_i(t - \tau_i(t))(1 - q_i(t))}{w_i(t)} - x_i(t - \tau_i(t)) q_i(t) \frac{w_i(t)}{2}. \tag{4.7}$$

The first term on the right side of Eq. 4.7 describes the additive window increment, and the second term stands for the multiplicative window decrement. The sending rate of TCP evolves according to

$$\dot{x}_i(t) \triangleq \frac{d}{dt}(\frac{w_i(t)}{\tau_i(t)}) = \frac{\dot{w}_i(t)}{\tau_i(t)} - \frac{w_i(t)\dot{\tau}_i(t)}{\tau_i^2(t)}. \tag{4.8}$$

The dynamics of the queue length $b_l(t)$ at link l can be expressed as

$$\dot{b}_l(t) = y_l(t) - C_l. \tag{4.9}$$

Denote w_i^*, p_l^*, q_i^* as the equilibrium values of $w_i(t)$, $p_l(t)$, $q_i(t)$, respectively. At equilibrium, $x_i^* = w_i^* \tau_i$, and $q_i^* = \sum_l R_{li} p_l^*$. Here, $\tau_i = d_i + \sum_l R_{li} \frac{b_l^*}{C_l}$ is the equilibrium round trip time for source i. It can be shown from Eq. 4.7 that

$$w_i^* = \sqrt{2(1 - q_i^*)/q_i^*}. \tag{4.10}$$

Let $\delta w_i(t) = w_i(t) - w_i^*$, $\delta p_l(t) = p_l(t) - p_l^*$, $\delta x_i(t) = x_i(t) - x_i^*$. Linearizing Eqs. 4.5, 4.7 and 4.8 around the equilibrium state with the first order approximation,

$$\delta \dot{w}_i(t) = -\frac{1}{\tau_i q_i^*} \sum_l R_{li} \delta p_l(t - \tau_{li}^b) - \frac{q_i^* w_i^*}{\tau_i} \delta w_i(t), \tag{4.11}$$

$$\delta\dot{x}_i(t) = \frac{\delta\dot{w}_i(t)}{\tau_i} - \frac{w_i^*\delta\dot{\tau}_i}{\tau_i^2}, \tag{4.12}$$

$$\delta\dot{y}_l(t) = \sum_i R_{li}\delta\dot{x}_i(t - \tau_{li}^f). \tag{4.13}$$

4.2 The VQR Scheme

VQR maintains a virtual queue $B_l(t)$, which is updated at each packet arrival according to

$$\dot{B}_l(t) = y_l(t) - \gamma C_l, \tag{4.14}$$

where γ is a positive number less than but close to 1 (e.g., 0.95). The marking probability is updated by

$$p_l(t) = \frac{g}{\gamma C_l}\{a(B_l(t) - B_{l,ref}) + (y_l(t) - \gamma C_l)\}, \tag{4.15}$$

where $B_{l,ref}$ is a reference virtual queue length, and g and a are positive numbers. If $p_l(t) < 0$, it is set to zero; if $p_l(t) > 1$, it is set to one.

At the equilibrium state, $\dot{B}_l(t) = 0$. Thus, the aggregate rate y_l^* is γC_l, which is slightly smaller than the link capacity C_l. Although this could lead to slight link under-utilization, there are some benefits of doing so. First, the queuing delay is zero at the equilibrium state. Second, since only small perturbation around the equilibrium state is considered, it can reasonably be assumed that $y_l(t)$ is always less than the link capacity. As a result, the queue length $b_l(t)$ is always zero, the round trip time $\tau_i(t)$ is reduced to d_i , and $\delta\tau_i(t)$ is always zero under such an assumption. Thus, the second term on the right side of Eq. 4.12 is negligible. Together with Eq. 4.11, Eq. 4.12 can be rewritten as

$$\delta\dot{x}_i(t) = \frac{\delta\dot{w}_i(t)}{\tau_i} = -\frac{1}{\tau_i^2 q_i^*}\sum_l R_{li}\delta p_l(t - \tau_{li}^b) - \frac{q_i^* w_i^*}{\tau_i}\delta x_i(t). \tag{4.16}$$

Linearizing Eq. 4.14 and 4.15 around the equilibrium state,

$$\delta\dot{B}_l(t) = \delta y_l(t), \tag{4.17}$$

$$\delta p_l(t) = \frac{g}{\gamma C_l}(a\delta B_l(t) + \delta y_l(t)). \tag{4.18}$$

If the similar procedure in [36] is followed, Eq. 4.13, 4.16, 4.17 and 4.18 can be expressed in the Laplace domain in the matrix form:

$$\delta \mathbf{y}(s) = \mathbf{R_f}(s)\delta \mathbf{x}(s), \tag{4.19}$$

$$\delta \mathbf{x}(s) = -(s\mathbf{I} + \mathbf{H_1})^{-1}\mathbf{H_2}\mathbf{R_b^T}(s)\delta \mathbf{p}(s), \tag{4.20}$$

$$\delta \mathbf{p}(s) = \mathbf{A}(s)\mathbf{G}\delta \mathbf{y}(s), \tag{4.21}$$

where $\mathbf{G} = diag\left(\frac{g}{\gamma C_l}\right)$, $\mathbf{A}(s) = diag\left(\frac{s+a}{s}\right)$, $\mathbf{H_1} = diag\left(\frac{q_i^* w_i^*}{\tau_i}\right)$, $\mathbf{H_2} = diag\left(\frac{1}{\tau_i^2 q_i^*}\right)$, and

$$[R_f]_{li} = \begin{cases} e^{-\tau_{li}^f s} & \text{if source } i \text{ uses link } l \\ 0 & \text{otherwise} \end{cases} \tag{4.22}$$

$$[R_b]_{li} = \begin{cases} e^{-\tau_{li}^b s} & \text{if source } i \text{ uses link } l \\ 0 & \text{otherwise} \end{cases} \tag{4.23}$$

Eqs. 4.19, 4.20 and 4.21 form a close loop control system with the return ratio

$$\mathbf{L}(s) = \mathbf{R_f}(s)(s\mathbf{I} + \mathbf{H_1})^{-1}\mathbf{H_2}\mathbf{R_b^T}(s)\mathbf{A}(s)\mathbf{G}. \tag{4.24}$$

According to [36,37], the above control system is stable if the eigenvalues of $L(j\omega)$ $(\omega \geq 0)$ do not encircle -1 in the complex plane. Define

$$\tilde{\mathbf{R}}(j\omega) = diag\left(\sqrt{\frac{g}{\gamma C_l}}\right)\mathbf{R_f}(j\omega)diag\left(\sqrt{x_i^*}\right). \tag{4.25}$$

Using the relationship

$$\mathbf{R_b}(s) = \mathbf{R_f}(-s)diag(e^{-\tau_i s}), \tag{4.26}$$

and following the similar argument in [37], the eigenvalues of $L(j\omega)$ are the same as those of:

$$\mathbf{Z}(j\omega) = diag\left(\frac{e^{-j\omega\tau_i}(j\omega+a)}{q_i^* x_i^*(j\omega\tau_i + q_i^* w_i^*)j\omega\tau_i}\right)\tilde{\mathbf{R}}^{\mathbf{T}}(-j\omega)\tilde{\mathbf{R}}(j\omega). \tag{4.27}$$

Denote E as the set of eigenvalues of $Z(j\omega)$. According to [37],

$$\begin{aligned} E \subset\ & \rho\left(\tilde{\mathbf{R}}^{\mathbf{T}}(-j\omega)\tilde{\mathbf{R}}(j\omega)\right) \times \\ & co\left(0 \cup \left(\frac{e^{-j\omega\tau_i}(j\omega+a)}{q_i^* x_i^*(j\omega\tau_i + q_i^* w_i^*)j\omega\tau_i}\right), i = 1,2,...,S\right), \end{aligned} \tag{4.28}$$

where $co(m_i, i = 1,2,...,S)$ denotes the convex hull of the set points $\{m_1, m_2, ..., m_S\}$, and the spectral radius of $\tilde{R}(j\omega)$ satisfies [37]:

$$\begin{aligned} & \rho\left(\tilde{\mathbf{R}}^{\mathbf{T}}(-j\omega)\tilde{\mathbf{R}}(j\omega)\right) \\ & \leq \left\| \mathbf{R_f^T}(-j\omega) diag\left(\frac{g}{\gamma C_l}\right) \mathbf{R_f}(j\omega) diag\left(x_i^*\right) \right\|_\infty \\ & \leq \left\|\mathbf{R_f^T}(-j\omega)\right\|_\infty \cdot \left\| diag\left(\frac{g}{\gamma C_l}\right) \mathbf{R_f}(j\omega) diag\left(x_i^*\right) \right\|_\infty = Mg, \end{aligned} \tag{4.29}$$

where M is the maximum number of links a TCP source traverses in the network.

Denote

$$\Lambda_i(j\omega) = \frac{Mg \times e^{-j\omega\tau_i}(j\omega+a)}{q_i^* x_i^*(j\omega\tau_i + q_i^* w_i^*)j\omega\tau_i}. \tag{4.30}$$

If $\Lambda_i(j\omega)(i = 1,2...M)$ do not encircle -1 in the complex plane, E does not either [37]. Therefore, the system is stable. Next, a sufficient condition to guarantee the stability of such a system will be presented.

Theorem 4.1 *The system is stable if* $a < \min_i \left\{\frac{1}{\tau_i}\sqrt{2q_i^*(1-q_i^*)};\ i = 1,...S\right\}$ *and* $g < \frac{\pi}{2M}\min_i\left(\sqrt{2q_i^*(1-q_i^*)}; i = 1,...S\right)$.

***Proof*:** From Eq. 4.30, $\Lambda_i(j\omega)$ is rewritten as

$$\Lambda_i(j\omega) = \frac{e^{-j\omega\tau_i}(j\omega\tau_i + a\tau_i)}{j\omega\tau_i(j\omega\tau_i + q_i^* w_i^*)} \frac{Mg}{q_i^* w_i^*}. \tag{4.31}$$

Using Eq. 4.10, it can be shown that the two conditions in Theorem 4.1 are equivalent to

$$a < \min_i \left\{ \frac{1}{\tau_i} q_i^* w_i^*; i = 1, ..., S \right\} \tag{4.32}$$

and

$$g < \frac{\pi}{2M} \min_i \left(q_i^* w_i^*; i = 1, ..., S \right). \tag{4.33}$$

Let

$$\frac{e^{-j\omega\tau_i}}{j\omega\tau_i} \frac{(j\omega\tau_i + a\tau_i)}{(j\omega\tau_i + q_i^* w_i^*)} = \rho e^{j\theta} \frac{e^{-j\omega\tau_i}}{j\omega\tau_i}, \tag{4.34}$$

where $\theta = phase\left(\frac{(j\omega\tau_i + a\tau_i)}{(j\omega\tau_i + q_i^* w_i^*)}\right)$ and $\rho = \left|\frac{(j\omega\tau_i + a\tau_i)}{(j\omega\tau_i + q_i^* w_i^*)}\right|$. If $a < \min_i \left\{ \frac{q_i^* w_i^*}{\tau_i}; i = 1, ..., S \right\}$, then $0 < \theta < \frac{\pi}{2}$ and $0 < \rho < 1$. Let ω' be any frequencies at which $\rho e^{j\theta} \frac{e^{-j\omega\tau_i}}{j\omega\tau_i}$ cross the negative real axis, then

$$-\omega'\tau_i - \frac{\pi}{2} + \theta = -(2k+1)\pi, \; k = 0, 1, 2, ... \tag{4.35}$$

Thus, from Eq. 4.34 and 4.35,

$$\frac{e^{-j\omega'\tau_i}}{j\omega'\tau_i} \frac{(j\omega'\tau_i + a\tau_i)}{(j\omega'\tau_i + q_i^* w_i^*)} = -\frac{\rho}{(2k + 1/2)\pi + \theta}. \tag{4.36}$$

From Eq. 4.35 and with the properties of ρ and θ ,

$$\frac{e^{-j\omega'\tau_i}}{j\omega'\tau_i} \frac{(j\omega'\tau_i + a\tau_i)}{(j\omega'\tau_i + q_i^* w_i^*)} > -\frac{2}{\pi}. \tag{4.37}$$

If $g < \frac{\pi}{2M} \min_i (q_i^* w_i^*; i = 1, ..., S)$, from Eq. 4.36,

$$\frac{e^{-j\omega'\tau_i}}{j\omega'\tau_i} \frac{(j\omega'\tau_i + a\tau_i)}{(j\omega'\tau_i + q_i^* w_i^*)} \frac{Mg}{q_i^* w_i^*} > -1. \tag{4.38}$$

In other words, $\Lambda_i(j\omega)$ only crosses the real axis on the right side of point -1, and thus never encircle -1 in the complex plane. Therefore, the system is stable. ■

4.3 Summary

In this chapter, VQR, a new AQM scheme based on the virtual queue length and the aggregate flow rate is proposed. VQR can achieve high link utilization and near zero queuing delay. It also presents and proves the local stability condition for a TCP/VQR based network with an arbitrary topology, heterogeneous round trip times, and backward propagation delays.

CHAPTER 5

EDGE BASED ACTIVE QUEUE MANAGEMENT (EAQM)

5.1 Introduction

Active Queue Management (AQM) has been a very active research area in recent years. Many AQM mechanisms have been proposed, e.g., random early detection (RED) [13], random exponential marking (REM) [17], PI controller [15], adaptive virtual queue (AVQ) [16], state feedback control (SFC) [18], and virtual queue and rate-based scheme (VQR) [45]. The main goals of these schemes are to reduce and stabilize queuing delay, avoid global synchronization, and achieve high link utilization. RED is a standard queue length based AQM that drops packets with probability proportional to the current average queue length. AVQ is a typical rate based scheme. REM, PI, and SFC use the queue length and the aggregate flow rate to compute the dropping probability. VQR uses both the virtual queue length and aggregate flow rate to determine the dropping probability.

However, AQM schemes have not been widely deployed in current Internet. One of the reasons is that all these AQM schemes need to be implemented at all routers (or at least bottleneck routers), and this demands significant upgrading to routers in the current networks. Another reason is that the performance of these AQM schemes have not been fully investigated and understood, and it is not clear which scheme performs the best. Therefore, it is still risky to deploy them in the entire network.

In this chapter, the framework of Edge-based AQM (EAQM) is proposed. EAQM is only deployed at the network edge and the Drop-Tail core routers are kept unchanged. Compared with traditional AQM schemes, it is able to provide comparable or even better performance; thus, it is more practical and economic. It is well known that TCP's throughput is inversely proportional to the round trip time (RTT); considerable unfairness

occurs to TCP connections with longer RTTs. As one of the key features, EAQM alleviates this type of unfairness, while current available AQM schemes fail to do so.

The remainder of this chapter is organized as follows. Section II presents the framework of EAQM. Section III presents the simulation results. Section IV presents the conclusions.

5.2 System Environment

In this chapter, the term "aggregate" is used to refer to all the TCP connections, which enter the network at the same ingress node and leave the network at the same egress node. Note that any edge node can be the ingress or egress node for multiple aggregates at the same time, and each aggregate can be identified with one pair of ingress and egress nodes uniquely. Each pair of ingress and egress nodes in each aggregate exchange information about the aggregate membership and network congestion condition by sending *feedback* packets to each other. At ingress nodes, packets are classified into aggregates. Based on the received congestion information, ingress nodes compute dropping probabilities for the corresponding aggregates, and drop packets based on the computed probabilities. The rest of this section will explain the feedback protocol between ingress and egress nodes, operations at ingress and egress routers, and how the dropping probabilities are computed.

A feedback protocol between ingress and egress nodes similar to [46] is adopted in this chapter. There are two types of feedback packets: *forward* packets, which are sent from ingress nodes to egress nodes, and *backward* packets, which are sent from egress nodes to ingress nodes. Both types of packets contain the timestamp field and aggregate information field. The timestamp field serves to calculate the RTTs between the two edge nodes, and the one-way queuing delay from the ingress node to the egress node. Based on the variation of RTTs, ingress nodes calculate dropping probabilities. Each ingress node fills in the aggregate information field with necessary information about the TCP flows that enter the network through itself. This allows the egress nodes to know at which ingress node

each TCP flow enters the network. Similarly, the aggregate information field in *backward* packets enables ingress nodes to know at which egress node each TCP flow leaves the network. Based on the information in *backward* packets, ingress nodes are able to know to which aggregate each TCP flow belongs, and this allows ingress nodes to classify TCP flows into the correct aggregates.

In order to calculate RTTs and one-way queuing delays, ingress nodes continue to send out forward packets after a certain time interval T. In this chapter, T for each aggregate is set to 100ms. The timestamp field in the *forward* packets is always available, but the aggregate information field is optional. The aggregate information field is needed only when a new flow enters the network through an ingress node, or one existing flow becomes inactive. When an egress node receives one *forward* packet, it immediately sends a *backward* packet back to the corresponding ingress node. The *backward* packet contains the original timestamp of the *forward* packet, and the time the corresponding *forward* packet was received. Again, the aggregate information field of the *backward* packet is not always required, and it is filled in only when the egress node detects new flows or an existing flow becomes inactive. Fig. 5.1 shows the architecture of an ingress node in EAQM.

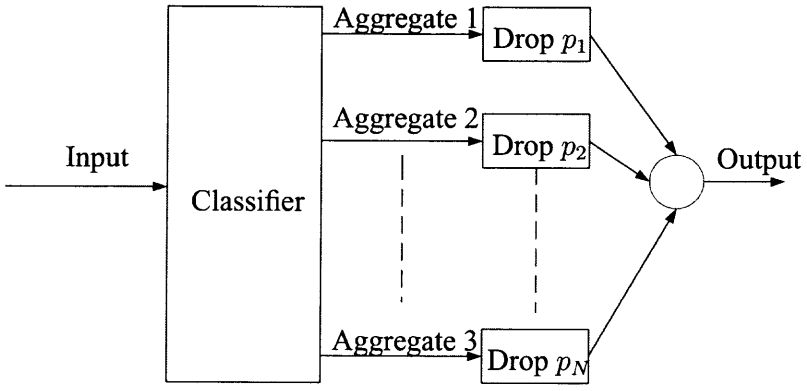

Figure 5.1: Architecture of an Ingress Node in EAQM.

Upon arrival of a *backward* packet, each ingress node uses its timestamp field to compute the RTT and one-way queuing delay for the corresponding aggregate. Then, the averaged RTT is updated:

$$RTT_{ave} \leftarrow (1 - w_{RTT})RTT_{ave} + w_{RTT}RTT \tag{5.1}$$

where w_{RTT} is the smoothing factor. One method to compute the one-way queuing delay q is to use $BaseRTT$ to track the minimum RTT, and compute q as $q = RTT - BaseRTT$. This approach is implemented in current TCP Vegas, but the performance degrades significantly when there is congestion in the reverse path from the egress node to the ingress node. This disadvantage was pointed out and the remedy was proposed in [47]. In this chapter, the same approach in [47] is adopted to compute the one-way queuing delay, and interested readers are referred to [47] for more details.

In EAQM, one simple approach to compute the dropping probability p is:

$$p \propto q_{ave}, \tag{5.2}$$

where q_{ave} is the average one-way queuing delay. There is similarity between this approach and RED. RED uses a linear function of the queue length at each router, and Eq. 5.2 indicates that p is proportional to the average one-way queuing delay q. The throughput of TCP can be expressed as:

$$R = \frac{1}{d\sqrt{\frac{2p}{3}} + RTO \min\left(1, 3\sqrt{\frac{3p}{8}}\right) p(1 + 32p^2)}, \tag{5.3}$$

where d is the round trip time, and RTO is the retransmission timeout, which is set to $4d$ in this chapter. When the dropping probability p is very small, Eq. 5.3 is reduced:

$$R = \frac{1}{d}\sqrt{\frac{3}{2p}}. \tag{5.4}$$

It can be seen that there is considerable unfairness against TCP flows with longer round trip time from both Eqs. 5.3 and 5.4. The following dropping probability is proposed:

$$\sqrt{\frac{1}{\beta q_{ave}}} = \frac{1}{RTT_{ave}\sqrt{\frac{2p}{3}} + 4RTT_{ave}\min\left(1, 3\sqrt{\frac{3p}{8}}\right)p(1+32p^2)}, \tag{5.5}$$

where β is a constant. The dropping probability p can be computed by solving Eq. 5.5. Since no explicit expression of p from Eq. 5.5 is available, p has to be numerically determined, and it leads to considerable computing overhead at each ingress node. Thus, the following approximation is adopted by replacing with $\min\left(1, 3\sqrt{\frac{3p}{8}}\right)$ and neglecting the p^2 term in Eq. 5.5:

$$\sqrt{\frac{1}{\beta q_{ave}}} = \frac{1}{RTT_{ave}\sqrt{\frac{2p}{3}} + 12RTT_{ave}\sqrt{\frac{3p}{8}}p}. \tag{5.6}$$

Eq. 5.6 can be reduced to a cubic equation of p, and p is set to be the unique positive root of Eq. 5.6. When p is very small, 5.6 can be simplified to

$$p = \frac{3\beta q_{ave}}{2RTT_{ave}^2}. \tag{5.7}$$

Substituting it into Eq. 5.4:

$$R = \sqrt{\frac{1}{\beta q_{ave}}} \times \frac{RTT_{ave}}{d}. \tag{5.8}$$

It is reasonable to assume that the propagation delay between TCP senders and the ingress node, and that between the egress node and TCP receivers are much smaller than the round trip time (RTT_{ave}) between the two edge nodes. Therefore, $\frac{RTT_{ave}}{d} \approx 1$, and Eq. 5.8 can be rewritten as:

$$R \approx \sqrt{\frac{1}{\beta q_{ave}}}. \tag{5.9}$$

Thus, the throughput unfairness against TCP flows with longer round trip time is eliminated. Simulations will be used to demonstrate this improvement in the next section.

In the next section, the performance of EAQM will be studied and compared with RED by simulations. It will be shown that EAQM can provide similar or even better performance as compared to RED in terms of queue length stability, responsiveness to traffic dynamics, and TCP throughput fairness. There are other possible ways in EAQM to compute the dropping probability other than using Eq. 5.6, e.g., using both queuing delay q and its changing rate $\dot{q}$. The focus of this chapter is to introduce the EAQM framework, which can potentially be adopted in enhancing current Internet, and will prompt more relevant research activities.

5.3 Simulations

A simple network shown in Fig. 5.2 is adopted in this section for simulations. The link from node N_1 to N_2 is the only bottleneck. There are two TCP aggregates, and each consists of 30 long-live TCP flows: the first aggregate is from S_1 to R_1, and the second aggregate is from S_2 to R_2. E_1, E_2 are ingress nodes, and E_3 and E_4 are egress nodes. The TCP packet size is 500 bytes, and the buffer size at N_1 is 800 packets. All simulations are 200 seconds long, and the throughputs are computed based on the last 100 seconds of the simulations.

The end of this chapter contains the guideline to set the parameters in EAQM to ensure stability. The parameters setting for EAQM are: $\beta = 0.015$, $w_{RTT} = 0.01$ and $T = 0.1$sec. The guideline in [15] is followed to set the parameters of RED to ensure it is stable: $min_{th} = 50$ packets, $max_{th} = 500$ packets, $p_{max} = 0.1$, and the weight w for queue averaging is 10^{-6}. All links except the bottleneck link are Drop-Tails. In all simulations, RED is used at the bottleneck link from N1 to N2 first; then EAQM is enabled at ingress and egress nodes ($E_1 ... E_4$,) and the bottleneck link from N_1 to N_2 is set back to simple Drop-Tail.

The first set of simulations compare the performance between RED and EAQM with respect to queue length stability. The evolution of the queue length at the bottleneck N_1

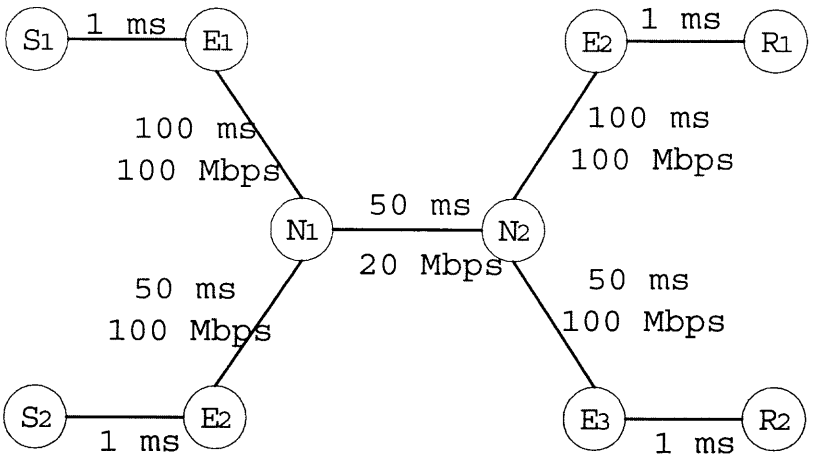

Figure 5.2: Simple Network Topology.

is plotted in Fig. 5.3. It can be seen that the instantaneous queue length under EAQM oscillates less and is more stable.

In the second set of simulations, the responsiveness of EAQM to traffic dynamics is investigated. The propagation delay between E_2 and N_1 and that between N_2 and E_4 are changed to 30ms. At time zero, 30 TCP connections start in the first aggregate, and no traffics are in the second aggregate. At the 100^{th} second, 30 TCP connections start in the second aggregate. The evolution of the queue length is plotted in Fig 5.4. It is very clear that the instantaneous queue length under RED fluctuates significantly between the 100^{th} and 140^{th} seconds, while EAQM is much less vulnerable to the traffic dynamics.

The third set of simulations are used to demonstrate that EAQM is immune to the throughput bias against TCP flows with longer round trip times. The propagation delay between E_1 and N_1 is kept identical to that between N_2 and E_3, and their values are changed from 10ms to 60ms. The fairness indices are calculated for both RED and EAQM. The fairness index F is defined as:

$$F = \frac{\min(r_1, r_2)}{\max(r_1, r_2)}, \tag{5.10}$$

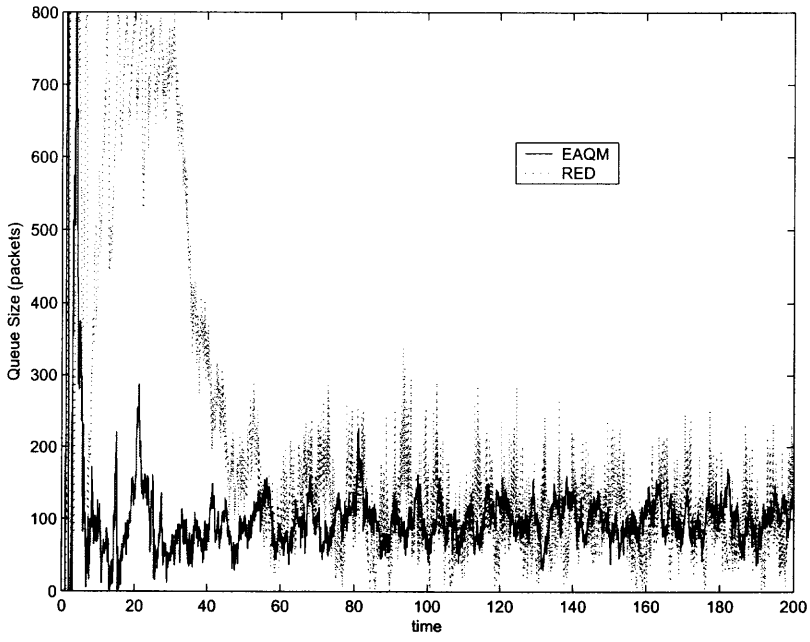

Figure 5.3: Queue Dynamics Comparison Between EAQM and RED.

where r_1 and r_2 are the throughput of aggregate one and two, respectively. The closer the index to one, the fairer among all TCP flows. The results are shown in Fig. 5.5. It can be seen that EAQM greatly reduces the unfairness.

Simulations based on the network with two bottleneck links shown in Fig.5.6 are also conducted. Each of S_1, S_2, and S_3 represents 30 TCP senders, and R_1, R_2, and R_3 are the corresponding TCP receivers. The link between N_1 and N_2 and the link between N_1 and N_2 are the bottlenecks. RED is used at bottlenecks first; then EAQM is enabled at ingress and egress nodes (E_1, ..., E_5,) and the bottleneck links are set back to simple Drop-Tail. All the RED and EAQM parameters are the same as before. The propagation delay between E_1 and N_1 is kept identical to that between N_2 and E_3, and their values are changed from 10ms to 60ms. The fairness indices for both RED and EAQM schemes are calculated. The fairness index F is defined as:

$$F = \frac{\min(r_1, r_2, r_3)}{\max(r_1, r_2, r_3)}, \tag{5.11}$$

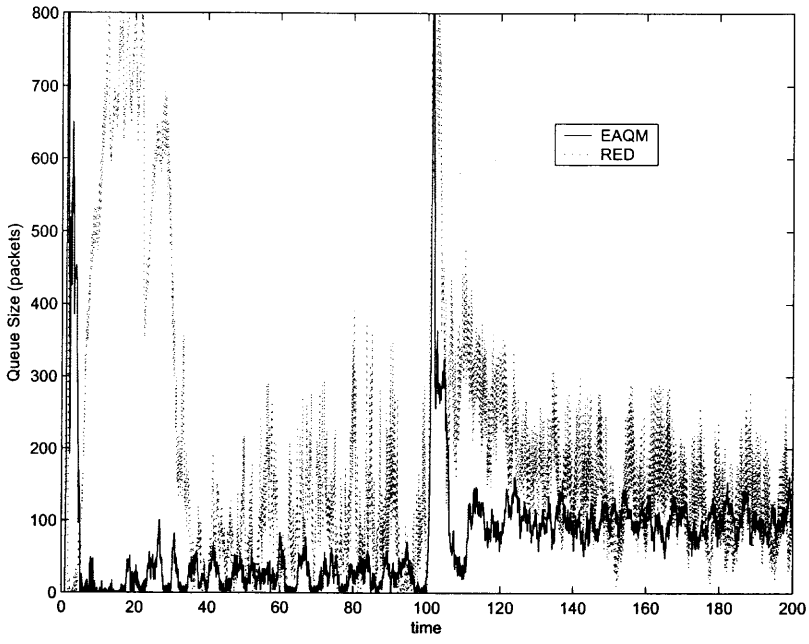

Figure 5.4: Queue Responsiveness Comparison Between EAQM and RED.

where r_1, r_2 and r_3 are the throughput of aggregate one, two and three, respectively. The results are shown in Figure 5.7.

Again, it is clear that the fairness index of EAQM is higher than that of RED. It can be seen that the fairness index of EAQM in Fig. 5.7 is around 0.7 instead of 1.0 as in Fig. 5.5. This is NOT because that EAQM does not eliminate the throughput unfairness against TCP with longer RTTs. It is because that the aggregate one (from S_1 to N_1) traverses two bottleneck links and each of the remainder two aggregates only traverses one bottleneck link. Thus, aggregate one experiences more queuing delay, and as a result, larger dropping probability than that of aggregate two and three. Owing to the symmetry, it can be known that the average queue lengths at each bottleneck are equal, and denote the value as Q_{ave}. From Eq.5.9, the throughput of aggregate one is

$$R_{1,eaqm} = \sqrt{\frac{1}{2\beta Q_{ave}}} \tag{5.12}$$

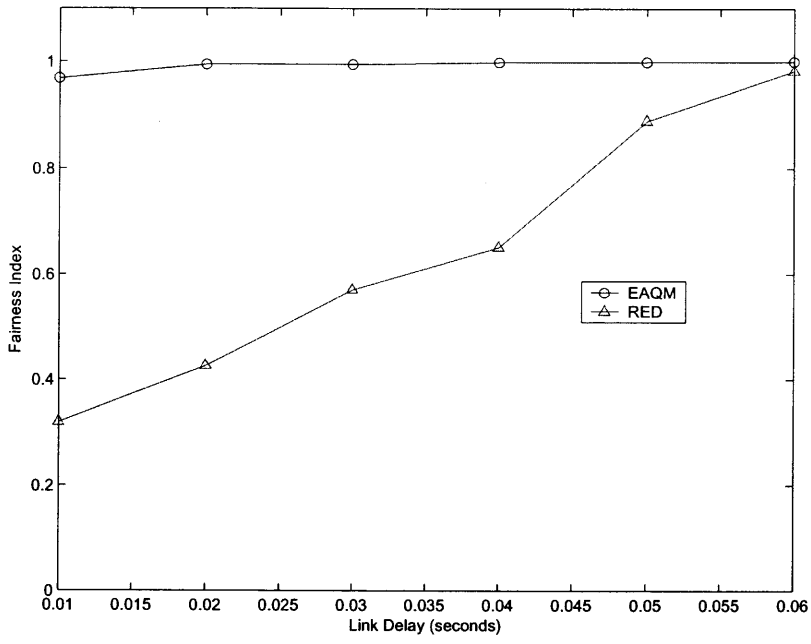

Figure 5.5: Fairness Index Comparison in Single Bottleneck Network.

and the throughput of aggregate one or two is

$$R_{2,eaqm} = R_{3,eaqm} = \sqrt{\frac{1}{\beta Q_{ave}}}. \tag{5.13}$$

The fairness index for EAQM is

$$F_{eaqm} = \frac{R_{1,eaqm}}{R_{2,eaqm}} = \frac{1}{\sqrt{2}} \approx 0.707, \tag{5.14}$$

and this is very close to the result shown in Fig. 5.7.

Similar situation occurs if RED is adopted. Due to the symmetry, the equilibrium dropping probabilities at each bottleneck link are the same, and are denoted as P_{ave}. Thus, the dropping probability for aggregate one is $2P_{ave}$, and P_{ave} for aggregate two and three. From Eq. 5.4,

$$R_{1,red} = \frac{1}{d_1}\sqrt{\frac{3}{4P_{ave}}}, \tag{5.15}$$

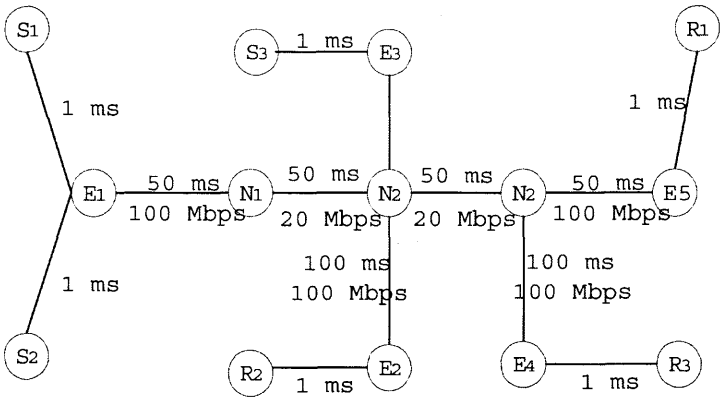

Figure 5.6: Multiple Bottlenecks Network.

$$R_{2,red} = \frac{1}{d_2}\sqrt{\frac{3}{2P_{ave}}} = R_{3,red} = \frac{1}{d_3}\sqrt{\frac{3}{2P_{ave}}}. \tag{5.16}$$

Here, d_1, d_2 $(= d_3 \leq d_1)$ and d_3 are the RTTs for aggregate one, two and three, respectively. The fairness index is

$$F_{red} = \frac{R_{1,red}}{R_{2,red}} = \frac{d_2}{d_1\sqrt{2}} \approx 0.707\frac{d_2}{d_1}. \tag{5.17}$$

Similarly, it can be shown that the factor $\frac{1}{\sqrt{2}}$ also exists under other traditional AQM schemes.

5.4 Summary

In this chapter, a new framework of AQM, namely, EAQM, is proposed. EAQM is only required to be deployed at the edge of the network. It has been shown that EAQM can achieve similar or better performance as compared to the typical AQM scheme RED. EAQM is a more practical and economic approach to improve current Internet.

5.5 Guideline to Set EAQM Parameters

The similar procedures in [48] can be used to set EAQM parameters. Considering N TCP connections with propagation delay d in a single bottleneck network. The linearized

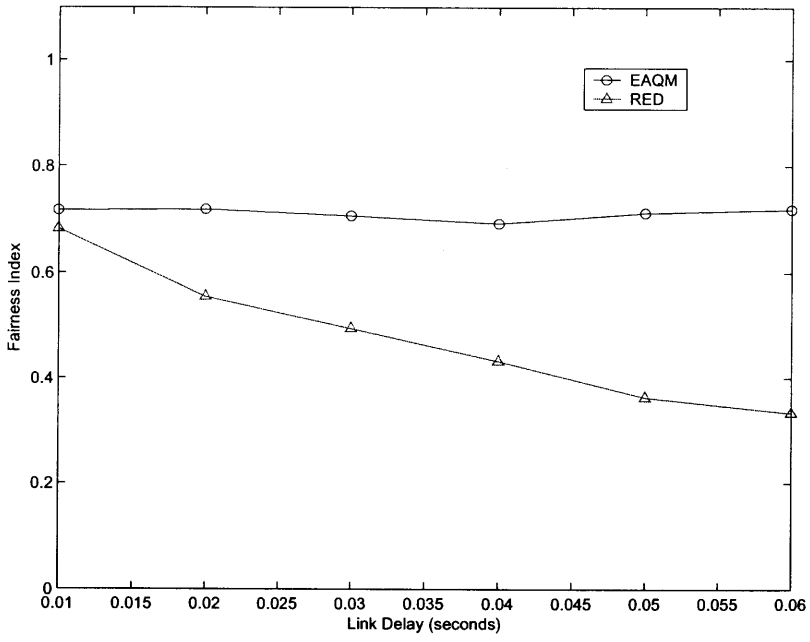

Figure 5.7: Fairness Index Comparison in Multiple Bottlenecks Network.

dynamic model in Laplacian domain is illustrated in Fig. 5.8. Readers are referred to [15,48] for more details on this model.

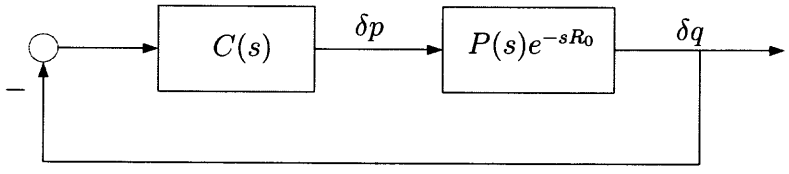

Figure 5.8: Linearized Control TCP/AQM Feedback System.

In Fig. 5.8, $P(s)$ is expressed as

$$P(s) = \frac{1}{C} P_{tcp}(s) P_{queue}(s), \tag{5.18}$$

where

$$P_{tcp}(s) = \frac{\frac{R_{ave}C^2}{2N^2}}{s + \frac{2N}{R_{ave}^2 C}}, \tag{5.19}$$

$$P_{queue}(s) = \frac{\frac{N}{R_{ave}}}{s + \frac{1}{R_{ave}}}. \tag{5.20}$$

Here, $P_{tcp}(s)$ and $P_{queue}(s)$ represent the dynamics of TCP and queue, respectively. Note that there is an additional term $\frac{1}{C}$ in Eq. 5.18 as compared to that in [48], and this is because q in this chapter is the queuing delay instead of the queue length used in [48]. From Eq. 5.7, $p = \frac{3\beta q_{ave}}{2RTT_{ave}^2}$,

$$\begin{aligned} \delta p &= \frac{3}{2RTT_{ave}^2}(1 - \frac{2q_{ave}}{RTT_{ave}})\delta q_{ave} \\ &= \frac{3}{2RTT_{ave}^2}(1 - \frac{2q_{ave}}{RTT_{ave}})\frac{1}{\frac{s}{K}+1}\delta q, \end{aligned} \tag{5.21}$$

where $K = \frac{-\ln(1-w_{RTT})}{T}$, w_{RTT} is the smoothing factor for RTT, and T is the time interval, after each of which the ingress nodes send out *forward* packets to probe one-way queuing delay for each aggregate. Combining Eqs. 5.18 and 5.21, the frequency response for the open loop transfer function is

$$L(j\omega) = \frac{3\beta RTT_{ave}\left(\frac{C}{2N}\right)^2(1 - \frac{2q_{ave}}{RTT_{ave}})e^{-j\omega R_{ave}}}{2\left(\frac{j\omega}{K}+1\right)\left(\frac{j\omega RTT_{ave}^2}{2N}+1\right)(j\omega RTT_{ave}+1)} \tag{5.22}$$

Using the result of Proposition 1 in [48], if $\frac{3}{2}\beta R^+\left(\frac{C}{2N^-}\right)^2 \leq \sqrt{\frac{\omega_g^2}{K^2}+1}$, where $\omega_g = 0.1\min\left\{\frac{2N^-}{(R^+)^2C}, \frac{1}{R^+}\right\}$, then the linearized feedback system is stable for $N \geq N^-$ and $R_{ave} \leq R^+$. According to the result of Proposition 2 in [48], the above feedback system is also stable if either $RTT_{ave} < 15R^+$ or $N > \frac{N^-}{5\pi}$.

In all simulations in this chapter, $N \leq 60$, and $RTT_{ave} < 0.6sec$ are used. Let $N^- = 60$ and $R^+ = 0.04sec$. Based on the above analysis, the linearized EAQM system is stable. The Nyquist plot of $L(j\omega)$ for $N \leq 60$, and $RTT_{ave} < 0.6sec$ is also shown in Fig. 5.9.

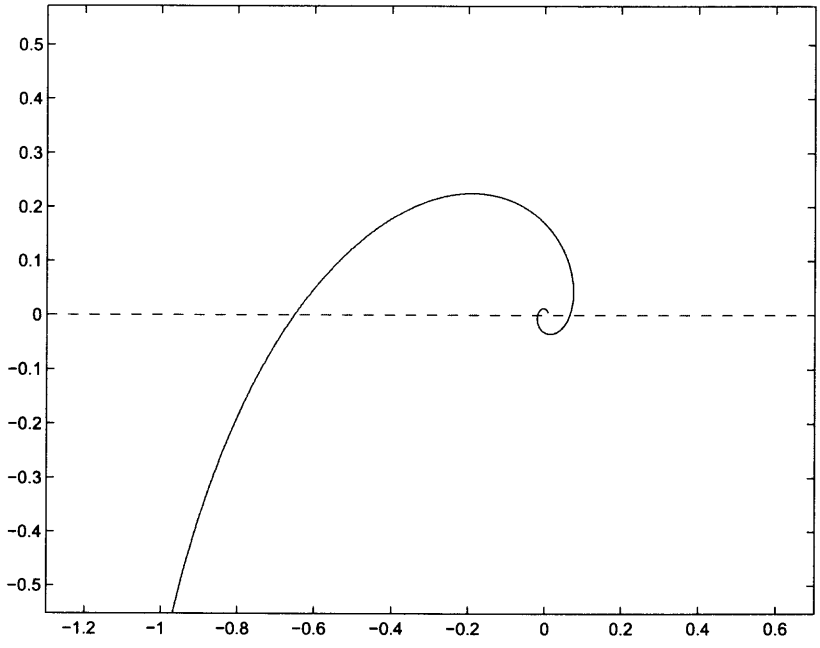

Figure 5.9: Nyquist plot of $L(j\omega)$, $N = 60$, $RTT_{ave} = 0.6sec.$

CHAPTER 6

FAIR BANDWIDTH ALLOCATION IN ASSURED FORWARDING

6.1 Introduction

The Differentiated Services (DiffServ) architecture is a promising means to provide QoS in the Internet. DiffServ provides service differentiation at the service class level instead of per flow level. In DiffServ, three service classes have been defined: Expedited Forwarding (EF), Assured Forwarding (AF), and Best Effort (EB). User traffics are classified into different service classes at the edge of the DiffServ network and receive corresponding services in the core of the network based on their service classes. Owing to its scalability and flexibility, DiffServ has drawn significant attention.

There are three key components in AF: Service Level Agreements (SLAs) between users and ISPs, packet markers at the network edges, and queue management in the core of the network. SLAs specify users' traffic profiles, which can be described by Commitment Information Rates (CIRs) and Peak Information Rates (PIRs). At network edges, packet markers are used to mark packets according to their conformance to their traffic profiles. Packets that conform to the SLAs are marked as "IN", and those do not as "OUT". In the core of the network, "IN" packets are given higher service priority than "OUT" packets. This type of service differentiation can be achieved by using RIO (RED with In/Out) [49], which drops "IN" packets with a probability lower than that of "OUT" packets.

Two packet marking models have been proposed for AF services: per-flow based and per-aggregate based. In the per-flow based model [50, 51, 52, 53], each individual flow has its own traffic profile and is marked by a dedicated packet marker at the network edges. In the per-aggregate based model [52, 54, 55, 56, 57], flows in the same aggregate are characterized with one traffic profile and marked by one marker. In this chapter, the

aggregate-based marking approach is adopted because SLAs between customers and ISPs are usually made at the aggregate level rather than the per-flow level.

In the context of AF services, a widely adopted fairness criterion of bandwidth allocation is that bandwidth should be allocated to aggregates (or flows if per-flow model is used) in proportion to their subscription rates (CIRs) [50, 51, 53, 52, 54, 55, 56, 57]. It has been shown that AF bandwidth is unfairly distributed in favor of traffics with smaller subscription rates [52]. Works in [59, 58, 56, 57] have shown that TCP round trip times (RTTs), subscription rates, and the number of flows in the aggregates can significantly affect the fairness.

Several schemes have been proposed to improve the fairness of AF bandwidth allocation. Authors in [50] proposed Adaptive Packet Marking (APM), which can achieve very good fairness for TCP traffics. However, APM needs to change the TCP scheme itself. Proposals in [51, 53] aimed to reduce the unfairness caused by the heterogeneity of RTTs and subscription rates. These two schemes are based on certain TCP throughput models, and they may not work well if end users adopt different protocols. Active Rate Management (ARM) was proposed in [55] to achieve fair bandwidth allocation among TCP aggregates. However, ARM has difficulty to guarantee fairness in both under-subscription and over-subscription cases. Authors in [54, 56] extended the idea of "TCP trunking" [60] to provide fair bandwidth allocation among aggregates. However, these studies have some shortcomings: first, they do not provide theoretical analysis to guarantee the fairness; second, they only perform simulations, in which each aggregate has the same target rate, and how these schemes work in a more general case remains unclear. An adaptive marking scheme was proposed in [57] to allocate bandwidth to aggregates in proportion to their target rates. Although simulations on a multiple bottleneck link network was provided in this work, it showed its effectiveness for only a specific case, and it may still have difficulty to assure fairness in a more general case.

Although these remedies can improve fairness from different aspects at different degrees, they have some common drawbacks. First, they do not provide fairness criteria for networks with multiple bottleneck links, since the fairness criterion that each aggregate receives bandwidth proportional to its target rate is only suitable to networks with single bottleneck link. Second, they did not provide solid theoretical analysis to show that the fairness can be guaranteed, especially in multiple bottleneck link networks.

In this chapter, the existing fairness criterion is extended to networks with multiple bottleneck links. Based on the extended fairness criterion, a novel adaptive packet marking scheme, Network-assist Packet Marking (NPM) is proposed. Both theoretical and experimental studies are used to show that NPM can achieve fair bandwidth allocation in both single and multiple bottleneck link networks, regardless of the RTT, subscription rate, and the number of flows of each aggregate.

The remainder of this chapter is organized as follows. Section II presents the necessary background. Section III presents the main ideas of NPM. Section IV presents the simulation results, followed by conclusions in Section V.

6.2 Background

Assume there are N TCP aggregates competing bandwidth in the network. Denote $R_{0,j}$, $R_{f,j}$ and R_j as the subscription rate, fair share rate, and the received rate of the j^{th} $(j = 1, 2, ..., N)$ aggregate, respectively. The packets in each aggregate are marked by one marker at the network edges. There are two types of packet makers: Time Sliding Window (TSW) marker characterized by target rate $R_{t,j}$, and Token Bucket (TB) marker characterized by target rate $R_{t,j}$ and bucket depth B_j. At the early stage of study on AF services, the target rate of the markers is set to the subscription rate of each aggregate, i.e., $R_{t,j} = R_{0,j}$. However, studies have shown that this approach leads to unfairness, and many remedies have been proposed [50,51,54,55,59,58]. The basic idea behind these new

approaches is to dynamically change the target rate $R_{t,j}$ (and B_i if TB is used) so that the fairness is improved.

Based on the fairness criterion introduced in Section I, the fair share rate of aggregate j in a single bottleneck network should be proportional to its subscription rate:

$$R_{f,j} = \frac{R_{0,j}}{\sum_{k=1}^{N} R_{0,k}} C, \tag{6.1}$$

where C is the bottleneck link bandwidth. If the received rate of each aggregate equals to its fair share, i.e., $R_j = R_{f,j}$, the fairness is achieved. In order to quantitatively describe the bandwidth allocation fairness, the following two definitions are introduced:

Definition 6.1 *The fairness ratio F_j of flow j is defined as the ratio between its received rate and its fair share rate:*

$$F_j = \frac{R_j}{R_{f,j}}. \tag{6.2}$$

Definition 6.2 *The fairness index F of the bandwidth allocation in the entire network is defined as:*

$$F = \frac{\min_j\{F_j\}}{\max_j\{F_j\}}. \tag{6.3}$$

Note that $F \leq 1$. The closer F is to one, the fairer it is.

In this chapter, RIO [49] is adopted in the core routers. In RIO, p_{in} and p_{out}, the dropping probability for "IN" and 'OUT" packets, are computed based on the average queue length q. Fig. 6.1 shows how p_{in} and p_{out} are computed. Interested readers are referred to [49] for more details.

6.3 Network-assist Packet Marking (NPM)

Before presenting NPM, the fairness criterion for single bottleneck networks should be extended to multiple bottleneck networks. The new fairness adopted in this chapter is weighted max-min fairness, in which the weight is the subscription rate $R_{0,j}$ for aggregate

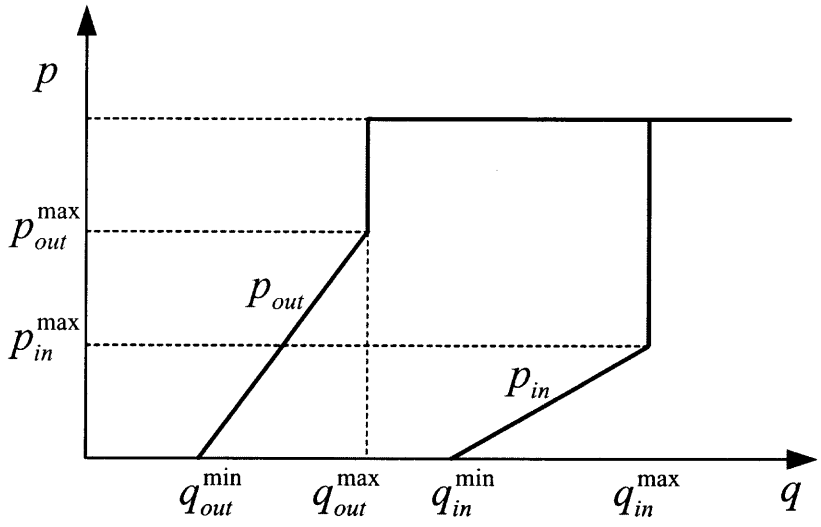

Figure 6.1: RIO Settings for AF Service.

j. It can be shown that weighted max-min fairness is reduced to (6.1) in the single bottleneck case. Therefore, weighted max-min fairness is a natural extension of the fairness described in Eq. 6.1.

In NPM, two steps are used to achieve this new fairness: first, the weighted max-min fair share rate for each aggregate is computed; second, after the fair share rate for each aggregate has been computed, each ingress edge node applies an adaptive marking algorithm to regulate each aggregate at the network edge so that its received rate equals to its fair share rate. The details of these two steps will be shown in the remaining of this section.

6.3.1 Computation of Fair Share Rate in NPM

Weighted max-min fairness is a straightforward extension of max-min fairness, and many algorithms used to compute max-min fair share rates can be modified to compute weighted max-min fair share rates. There are two types of algorithms to compute max-min fair share rates: global synchronized algorithm and distributed asynchronous algorithm. The global synchronized algorithm [61] requires global knowledge of the network, and it is impractical to be implemented in the current network. In the distributed asynchronous algorithm, the

fair share rates can be computed without global information. This type of algorithm is adopted in this chapter.

There have been extensive studies on the distributed asynchronous algorithm in the context of ATM networks [62]. In this chapter, the idea in [63] is borrowed to compute the fair share rates. In [63], each node keeps the information of the crossing flows, and computes the fair share rate for each of them in a distributed manner. In the context of the AF service with aggregate marking model, only the fair share rate of each aggregate needs to be computed. Thus, each node only needs to keep the information of the crossing "aggregates" instead of the flows. The number of aggregates is much less than the number of flows, and keeping per aggregate information at the core nodes becomes affordable. Therefore, it is reasonable and practical to borrow the idea in [63] at the aggregate level. Next, the method in NPM to compute the fair share rates will be presented.

In order to compute fair share rates, the ingress edge nodes and the egress edge nodes need to communicate by periodically sending special control packets to each other. There are two types of feedback packets: *forward* packets, which are sent from ingress nodes to egress nodes, and *backward* packets, which are sent from egress nodes to ingress nodes. The control packet contains four fields: aggregate ID field, aggregate weight field, stamped rate field, and underloading field. Based on the ID field in each control packet, core nodes are able to identify all the crossing aggregates. The aggregate weight field is set to the subscription rate $R_{0,j}$ for the j^{th} aggregate, and this value is used as the weight to compute the weighted max-min fair share rate for aggregate j. The stamped rate field and the underloading field are used to compute the fair share rates. When the ingress edge node sends out a new *forward* packet, the value of the underloading field is set to zero. The purposes and usage of these two fields will be clearer later on in this section.

Each node keeps the "advertised rate" μ_j for aggregate j that crosses it, and μ_j represents the available bandwidth for aggregate j. Upon receiving a control packet for aggregate j, the node compares the "stamped rate" in the stamp rate field with the

"advertised rate". If the "advertised rate" is lower than the "stamped rate", the stamp rate field is changed to the "advertised rate" and the underloading field is set to one. Otherwise, the control packet is unchanged. When the egress edge node receives the control packet, it sends it back as a *backward* packet to the corresponding ingress edge node. After receiving this *backward* packet, if the value of the underloading field is one, the ingress edge node sets the "stamped rate" in the next outgoing control packet to the "stamped rate" of the received feedback packet. Otherwise, the "stamped rate" in the next outgoing *forward* packet is set to infinity.

Each node has a list of the crossing aggregates, and keeps track of their latest seen stamped rates, which are called "recorded rates". It also keeps another two lists: unrestricted list U and the restricted list R. U includes aggregates whose recorded rates are higher than their "advertised rates", and the remaining aggregates are in the set R. Assume that $j \in U$. Upon the reception of a new control packet from j, the "advertised rate" for j is recomputed as:

$$\mu_j = \frac{C - C_R}{\sum_{k \in U} R_{0,k}} R_{0,j}, \tag{6.4}$$

where C_R is the sum of all the recorded rates of the aggregates in the restricted list R, and C is the link capacity. Note that Eq. 6.4 is different from the procedure adopted in [63], since what is needed to compute is the weighted max-min fair rate instead of the max-min fair rate.

One thing worthy of mentioning is how frequently each ingress edge node sends out *forward* packets. In this chapter, each ingress edge node sends out a *forward* packet every round trip time (RTT) for each aggregate that is marked at this node. The authors in [63] have shown that the upper bound of the convergence time of the above algorithm is $4ND$, where D is the upper bound of all RTTs and N is the number of iterations for the global synchronized algorithm to converge. In fact, the convergence time is significantly smaller

than the upper bound. Therefore, after a limited time, the network can finish computing the fair rates.

6.3.2 Adaptive Marking in NPM

In this chapter, the Time Sliding Window (TSW) type of marker is adopted. At the beginning, the marker's target rate $R_{t,j}$ is initialized to the subscription rate $R_{0,j}$. Each ingress node uses a variable $R'_{f,j}$ to track the stamped rate in the *backward* packet. Upon the arrival of a new *backward* control packet, if the stamped rate in this packet is different from $R'_{f,j}$, $R'_{f,j}$ is updated to the new value, and the target rate ($R_{t,j}$) of the marker is also updated to the new $R'_{f,j}$

$$R_{t,j} = R'_{f,j}. \tag{6.5}$$

It is known that $R'_{f,j}$ will converge to the fair rate $R_{f,j}$ within $4ND$. Therefore, $R_{t,j}$ will be stabilized at the fair share rate $R_{f,j}$. This marking scheme is called the Simple Network-assist Packet Marking (S-NPM). When all $R_{t,j}$ is stabilized, all the bottleneck links in the network are exactly provisioned. In other words, the bandwidth of each bottleneck link is equal to the sum of the target rates of all aggregates crossing it. In this case, the received rate of aggregate j is [57]

$$R_j = \frac{(1-p_{in})R_{f,j}}{2}\left[1+\sqrt{1+\left(\frac{\sqrt{6}N_j}{(1-p_{in})R_{f,j}T_j}\right)^2}\right], \tag{6.6}$$

where p_{in} is the dropping probability of "IN" packets, N_j is the number of TCP flows in aggregate j, and T is the round trip time of aggregate j. It can be reasonably assumed that p_{in} is very small. If $R_{f,j}T_j/N_j \gg 1$,

$$R_j \approx R_{f,j}, \tag{6.7}$$

implying that the received rate for each aggregate is approximately equal to its fair share rate. Thus, S-NPM can guarantee fair bandwidth allocation under the above assumption.

However, $R_{f,j}T_j/N_j \gg 1$ is not always true, e.g., $N_j = 20$, $T_j = 0.05sec$, $R_{f,j} = 625$ packets/sec $= 5$Mbps (assume 1000 bytes per packet), $R_{f,j}T_j/N_j = 1.55 \not\gg 1$. Therefore, S-NPM sometimes may not guarantee fairness. In order to overcome the disadvantage of S-NPM, Enhanced Network Packet Marking (E-NPM) is proposed.

In E-NPM, each egress edge node monitors the received rate R_j of the aggregate and feeds it back to the corresponding ingress node by attaching it to the *backward* packet. At the arrival of the *backward* packet, each ingress edge node checks the stamped rate. If it is different from the old value of $R'_{f,j}$, the target rate ($R_{t,j}$) of the marker is updated according to Eq. 6.5. Otherwise, $R_{t,j}$ is updated as:

$$\begin{cases} \text{if } R_j \geq (1+\delta)R'_{f,j} & R_{t,j} \rightarrow (1-\beta)R_{t,j} \\ \text{if } R_j \leq (1-\delta)R'_{f,j} & R_{t,j} \rightarrow (1+\beta)R_{t,j} \\ \text{otherwise} & R_{t,j} \rightarrow R_{t,j} \end{cases} \tag{6.8}$$

Here, δ and β are two positive numbers, which are very close to zero. Since $R'_{f,j}$ converges to $R_{f,j}$ very quickly, Eq. 6.8 will finally become:

$$\begin{cases} \text{if } R_j \geq (1+\delta)R_{f,j} & R_{t,j} \rightarrow (1-\beta)R_{t,j} \\ \text{if } R_j \leq (1-\delta)R_{f,j} & R_{t,j} \rightarrow (1+\beta)R_{t,j} \\ \text{otherwise} & R_{t,j} \rightarrow R_{t,j} \end{cases} \tag{6.9}$$

Corollary 6.1 *When the E-NPM scheme is stabilized, the fairness index of the bandwidth allocation in the entire network is*

$$1 \geq F \geq \frac{1-\delta}{1+\delta}. \tag{6.10}$$

Proof: From Eq. 6.9, it is known that when E-NPM is stabilized, the target rate $R_{t,j}$ for the marker is also stabilized. Thus, the fair ratio of aggregate j satisfies:

$$\frac{1}{1-\delta} \leq F_j \leq \frac{1}{1+\delta}. \tag{6.11}$$

Based on the definition of *fairness index*

$$F = \frac{\min_j\{F_j\}}{\max_j\{F_j\}}, \tag{6.12}$$

and combining Eqs. 6.11 and 6.12,

$$1 \geq F \geq \frac{1-\delta}{1+\delta}. \tag{6.13}$$

This completes the proof. ■

Proposition 6.1 indicates that the fairness index under E-NPM scheme is very close to one, i.e., E-NPM can fairly distribute bandwidth among aggregates. In the next section, simulations will demonstrate E-NPM's capability to provide fair bandwidth allocation.

6.4 Simulations

In this section, simulation results in both single and multiple bottleneck link networks are presented. RIO [49] is adopted in the core routers to provide service differentiation between "IN" and "OUT" packets. The parameters for RIO are set as follows: $(q_{out}^{min}, q_{out}^{max}, p_{out}^{max}) = (50, 150, 0.1)$ and $(q_{in}^{min}, q_{in}^{max}, p_{in}^{max}) = (150, 500, 0.02)$. Both α and β in E-NPM are set to 0.01.

6.4.1 Single Bottleneck Link Network

This set of simulations study the performance of the proposals in a network with one bottleneck link, which is shown in Fig. 6.2. The link from core router C_1 to core router C_2 is the only bottleneck, and $E_1 - E_4$ are the edge routers. There are two TCP aggregates, each of which consists of 20 long-lived TCP flows by default. The first aggregate is from S_1 to R_1, and the second aggregate is from S_2 to R_2. The TCP packet size is 1000 bytes.

First, performance of S-NPM, E-NPM, and the original TSW are compared. The subscription rate $R_{0,1}$ of aggregate 1 is fixed to $10Mbps$, and the subscription rate $R_{0,2}$ of aggregate 2 varies between $1Mbps$ and $30Mbps$. The fairness indices (computed based on

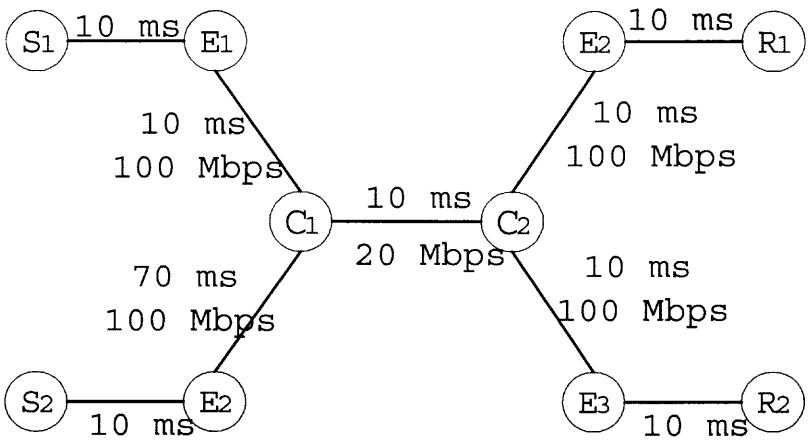

Figure 6.2: Single Bottleneck Link Network.

Eqs. 6.2 and 6.3) under these schemes are shown in Fig. 6.3. It is clear that both S-NPM and E-NPM have better performance than the original TSW. When $R_{0,2}$ is between $1Mbps$ and $5Mbps$, the fairness index of S-NPM is significantly smaller than one, demonstrating that S-NPM sometimes has difficulty to guarantee fairness. The reason is that the assumption $R_{f,j}T_j/N_j \gg 1$ and Eq. 6.7 do not always hold. On the other hand, the fairness index of E-NPM is always very close to one, demonstrating that E-NPM can guarantee fair bandwidth allocation all the time. In the remaining of this chapter, the focus is on the simulation study on the performance of E-NPM.

Next, effect of RTT of TCP aggregates on the performance of E-NPM is studied, and results are shown in Fig. 6.4 and Fig. 6.5. The propagation delay between E_1 and C_1 is fixed to $50ms$, and the propagation delay between E_2 and C_1 is varied between $10ms$ and $150ms$. There are two cases: under-subscription and over-subscription. In the under-subscription case, the subscription rates for aggregate one and two are fixed to $(5Mbps, 10Mbps)$. According to the fairness criterion, the fair rates are $(6.667Mbps, 13.333Mbps)$, which are shown by two solid horizontal lines in Fig. 6.4. In the over-subscription case, the subscription rates for aggregate one and two are fixed to $(40Mbps, 10Mbps)$, and the corresponding fair rates are $(16Mbps, 4Mbps)$, which are also shown by two solid horizontal

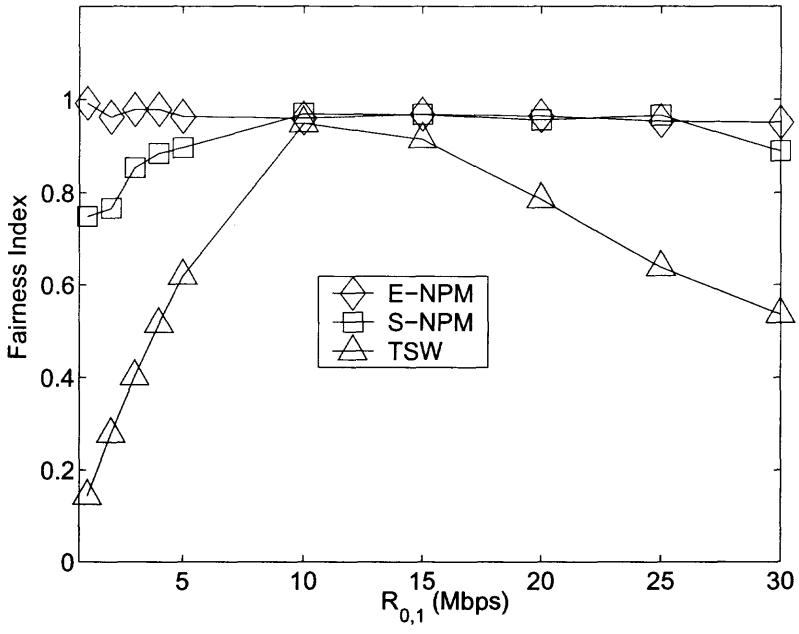

Figure 6.3: Comparison of Fairness for S-NPM, E-NPM and TSW.

lines in Fig. 6.5. From Fig. 6.4 and Fig. 6.5, it can be seen that in both cases, the received rates for two aggregates are very close to their fair share rates, and this demonstrates that RTT has very little impact on the performance of E-NPM.

How the number of TCP flows in the aggregate can affect the performance of E-NPM is also investigated. Two cases are considered: under-subscription and over-subscription. There are 20 TCP flows in aggregate 1, and the number of TCP flows in aggregate 2 varies between 20 and 80. The subscription rates for aggregates 1 and 2 are the same as those in the simulations for Fig. 6.4 and Fig. 6.5. Fig. 6.6 and Fig. 6.7 show the results. Again, it can be seen that E-NPM can achieve fair rate allocation regardless of the number of TCP flows in the aggregate.

6.4.2 Multiple Bottleneck Link Network

This simulation studies the performance of E-NPM in the multiple bottleneck link network, which is shown in Fig. 6.8. $C_1 - C_3$ are core routers, and $E_1 - E_5$ are the edge routers.

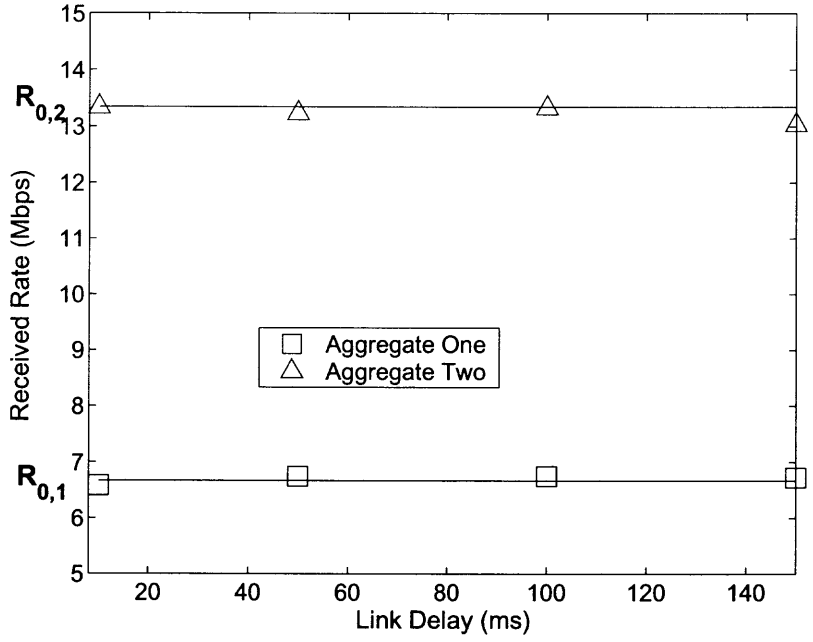

Figure 6.4: The Impact of RTT on E-NPM (under-subscription).

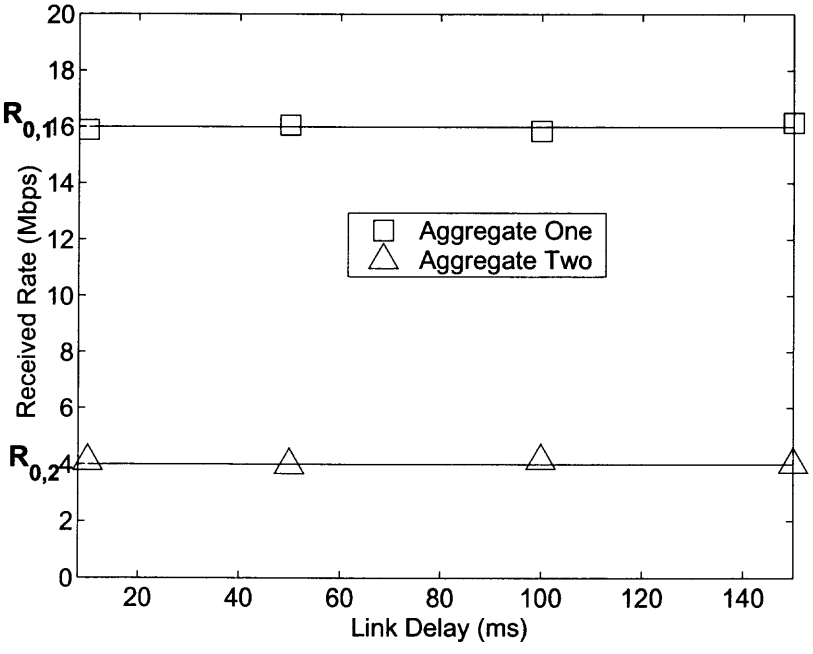

Figure 6.5: The Impact of RTT on E-NPM (under-subscription).

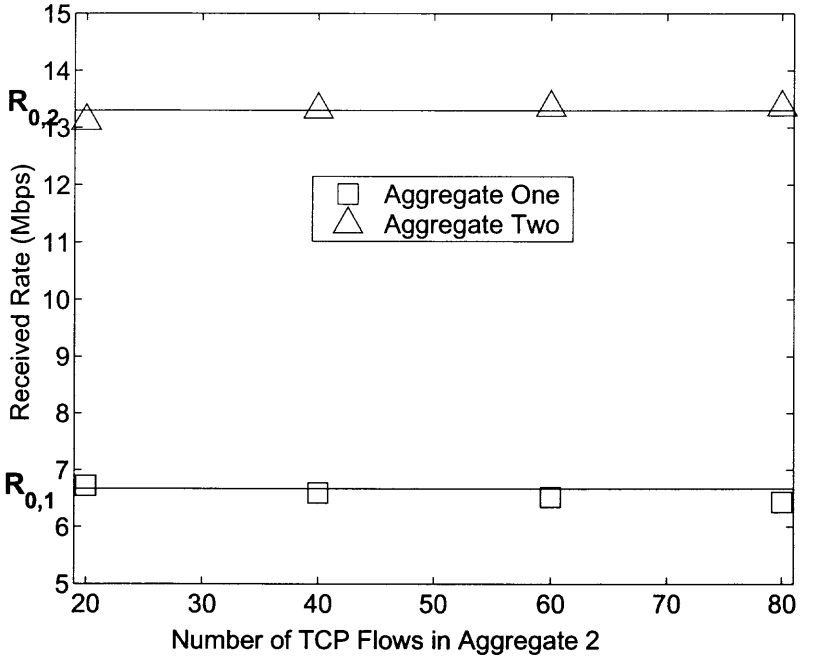

Figure 6.6: The Impact of Number of TCP Flows on E-NPM (under-subscription).

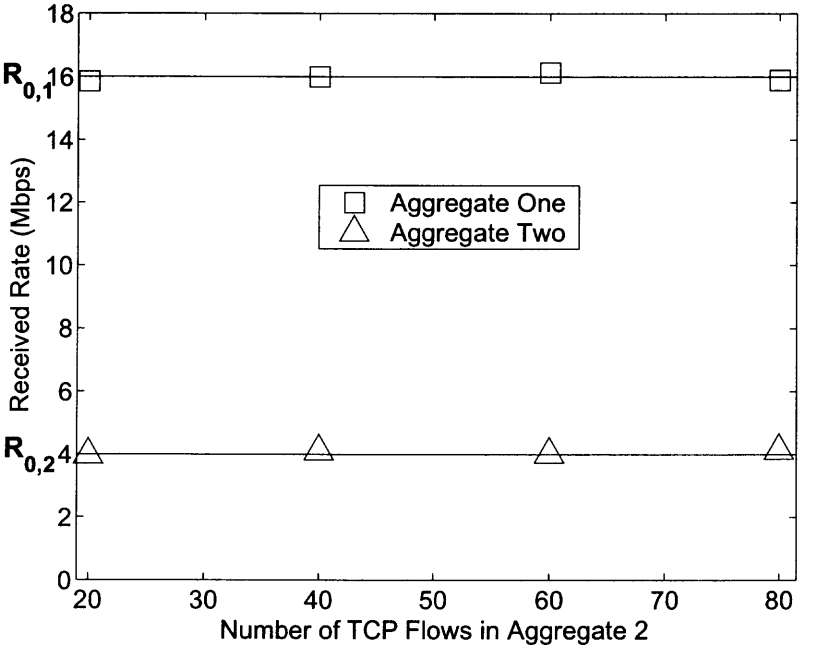

Figure 6.7: The Impact of Number of TCP Flows on E-NPM (over-subscription).

The link between C_1 and C_2 and the link between C_2 and C_3 are the bottlenecks. There are three TCP aggregates, each of which consists of 20 TCP flows. These aggregates start at (S_1, S_2, S_3) and end at (R_1, R_2, R_3), respectively. The subscription rates for aggregate 2 and 3 are fixed to $5Mbps$ and $10Mbps$, respectively. The subscription rate of aggregate 1 ($R_{0,1}$) varies between $1Mbps$ and $30Mbps$.

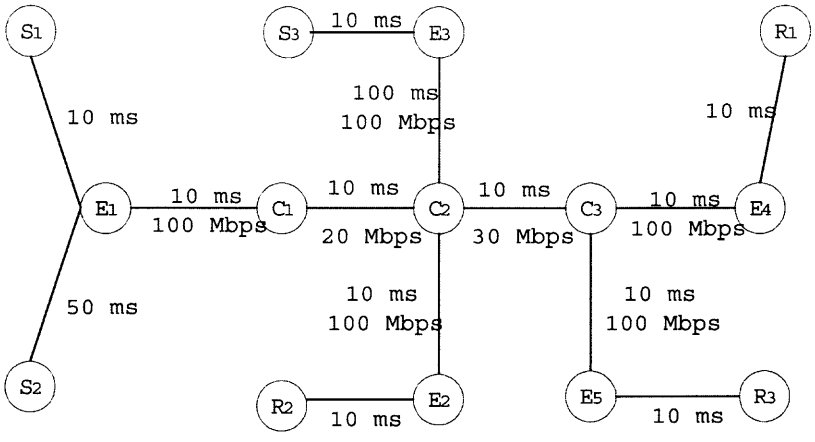

Figure 6.8: Multiple Bottleneck Link Network.

According to the extension of the fairness criterion in the multiple bottleneck link network, the fairness criterion in this simulation is the weighted max-min fairness, in which the weight for aggregate j is its subscription rate $R_{0,j}$. Fig. 6.9 shows the simulation results. It can be seen that E-NPM achieves good fairness performance in the network with multiple bottleneck links.

6.5 Summary

In this chapter, a new packet marking scheme called NPM (S-NPM and E-NPM) is introduced. NPM can be used for AF services in the DiffServ networks. NPM is based on per aggregate marking, and it only introduces a slight computation overhead in the core routers. The fairness criterion of bandwidth allocation is extended from the single bottleneck case to that of the multiple bottleneck case. From both theoretical analysis and experimental studies, it has been shown that both S-NPM and E-NPM significantly

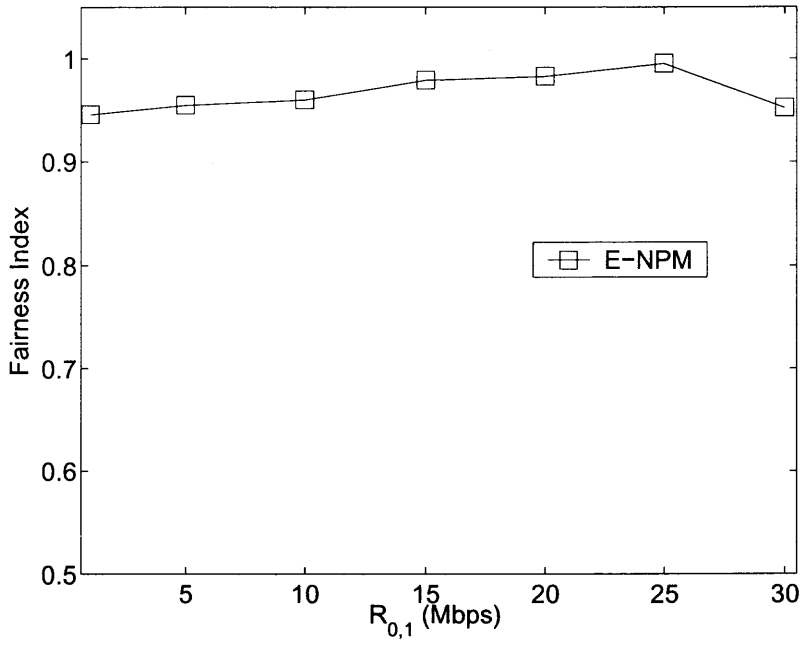

Figure 6.9: E-NPM Performance in Multiple Bottleneck Link Network.

outperform the original TSW packet marking scheme. Moreover, E-NPM can assure fair bandwidth allocation in both single and multiple bottleneck networks, and its performance is not affected by the subscription rate, round trip time, and number of flows in each aggregate.

CHAPTER 7

HIGHSPEED TCP IN OPTICAL BURST SWITCHING NETWORKS

There has been an increasing demand for reliable data transmission at high speed (1 Gbps or even 10 Gbps) from many applications in nuclear physics, telemedicine, etc. TCP is the most dominant transport layer protocol for data transmission in the present Internet. However, the current standard TCP cannot perform efficiently in the high-speed and long delay network environment. This inefficiency arises from its window adaptation approach in the congestion avoidance phase: TCP increases its window size by one every round trip time (RTT) if no packet is lost, and decreases its window size by half in response to a loss event. For example, in order to achieve 10 Gbps throughput for a standard TCP connection with 1500 bytes packets and 100 ms RTT, it requires an average window size of 83,333 and no packet loss in more than one hour [38]. In other words, it needs at most one loss event every 5×10^9 packets, which is beyond the bit error rate for the current network [38].

HighSpeed TCP (HSTCP) [38], FAST TCP (FAST) [24] and other new versions of TCP have been proposed to attain high throughput in the high bandwidth and long delay environment. HSTCP modifies the standard TCP with a more aggressive window increase and more conservative window decrease approach. HSTCP is the focus of this chapter, and a brief introduction to HSTCP will be presented later in this chapter. FAST adapts its window size based on the packet delay, and the study of FAST is beyond the scope of this work.

Optical Burst Switching (OBS) [39] provides a feasible paradigm for Internet over wavelength- division multiplexing (WDM) integration. Owing to its flexibility in achieving high utilization of optical bandwidth and its capability to support transparent data transmission, OBS is a promising technology to support future Internet backbone.

The performance of TCP over OBS has drawn attention of the research community. In [40], the throughput of TCP Reno over OBS is obtained. In [41], the effect of false Time Out (FTO) of TCP over OBS was investigated, and a new Burst TCP (BTCP) was proposed to detect FTO and react properly so that the throughput of BTCP was improved significantly over the standard TCP. All these works only considered two extreme cases, in which either all the TCP packets in one window are in the same optical burst or no packets are in the same optical burst. However, TCP with high transmission rate (such as HSTCP) does not fall into these two extremes. On the contrary, packets in one window usually spread over multiple optical bursts, and one optical burst can also contain multiple packets from one TCP connection. This chapter focuses on this scenario and derive the throughput upper bound of HSTCP over OBS.

The remainder of this chapter is organized as follows. Section I presents the brief introduction to HSTCP and OBS. Section II presents the analysis on the throughput of HSTCP over OBS, and provides an upper bound. Section III presents the conclusions.

7.1 Backgroud

The basic concept behind HSTCP is to use the following scheme to update the window size w [38]:

$$\begin{cases} w \leftarrow w + a(w) & \text{ACK} \\ w \leftarrow (1 - b(w))w & \text{otherwise} \end{cases} \tag{7.1}$$

where

$$a(w) = \begin{cases} High\_Window^2 \times High\_P^2 \times \frac{2b(w)}{2-b(w)} & \text{if } w > W \\ 1 \leftarrow (1 - b(w))w & \text{otherwise} \end{cases} \tag{7.2}$$

and

$$b(w) = \begin{cases} 0.5 + \frac{(High\_Decrease-0.5)\times(\log(w)-\log(W))}{\log(w_1)-\log(W)} & \text{if } w > W \\ 0.5 \leftarrow (1 - b(w))w & \text{otherwise} \end{cases} \tag{7.3}$$

Here, $High\_Decrease$, W, w_1, $High\_Window$ and $High\_P$ are predefined parameters. From Eq. 7.1, it can be seen that HSTCP still adopts Additive Increase Multiplicative Decrease (AIMD) window updating scheme as the traditional TCP does, except that $a(w)$ and $b(w)$, the increase and decease parameters are not constant any more. Note that $a(w)$ and $b(w)$ are increasing and decreasing functions of the window size w, respectively. In other words, HSTCP uses a faster window increase and slower window decrease scheme as the window size increases. W is set to 38 by default, corresponding to the throughput of 4.5Mbps for the standard TCP with 100ms round trip time and 1500 bytes packet size. From the expression of $a(w)$ and $b(w)$, it can be seen that when the window size is not larger than W, HSTCP reduces to the standard TCP. This ensures that HSTCP is fair to the standard TCP when the link capacity is small. Readers are referred to [38] for the reasons to use the above approach and the recommended values for these predefined parameters.

There are two parts in an optical IP over OBS network: local electronic IP networks and OBS backbone network. Edge nodes (ENs) reside between the electronic and optical world, and perform electronic-optical and optical-electronic conversions. At ingress ENs, each burst is assembled with multiple packets. In this chapter, it is assumed that the burst assembly time T_b is constant. After burstification and other necessary procedures, the burst is transparently transmitted in the optical backbone. Owing to the contention among the bursts inside the OBS network, bursts can be lost before they reach their destined egress ENs. At egress ENs, each burst is dissembled and the IP packets in the burst are forwarded to local electronic networks.

7.2 Model and Analysis

Consider a simplified network model shown in Figure 7.1. Assume the link between the TCP sender and ingress EN is C, the burst loss ratio inside the OBS network is p, the burst assembly time is T_b, and the round trip time of the TCP connection is RTT. Here, RTT includes the end-to-end propagation delay, buffering delay at the local IP network, and all

the optical processing delays. It is reasonable to assume that $RTT \gg T_b$. Intuitively, two factors can limit the throughput of the TCP connection: the values of C and p. If p is small, TCP throughput is constrained by the link capacity C. If p is large enough, TCP has to drop its window size very often, and it cannot make use of the link capacity C efficiently. It will be shown later in this chapter that the burst assembly time T_b also plays an important role in determining the throughput of TCP.

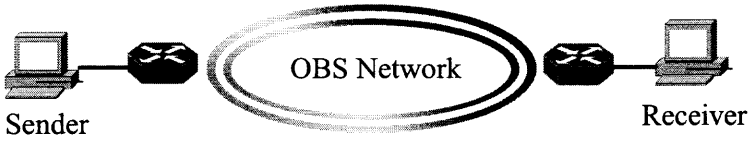

Figure 7.1: Simple Network Topology with HSTCP over OBS.

TCP tends to send packets back-to-back in a train due to its "ack-clocking" property (here the term burst is not used to avoid confusion with the burst in the OBS network). A common packet transmission pattern of a TCP connection is shown in Figure 7.2, in which the time is divided into RTT slots and each RTT slot contains one packet train.

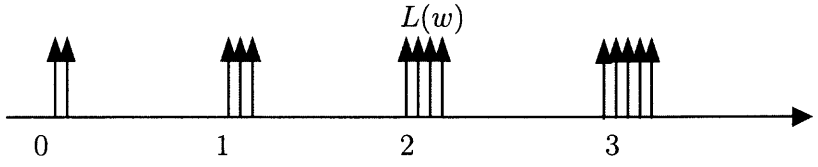

Figure 7.2: Illustration of Packet Pattern Without Pacing.

The length of each packet train increases as the window size increases. When the link capacity C is fully utilized, each RTT time slot will be entirely covered by one TCP packet train. When the window size is w, the length of one packet train in every RTT time slot is $L(w) = \frac{w}{C}$ and the packets sent in one RTT appear in $\lceil L(w)/T_b \rceil$ optical bursts, where $\lceil \bullet \rceil$ is the ceiling operator. Thus, the probability that TCP packets get lost due to the optical burst loss in one RTT is:

$$q(w) = 1 - (1-p)^{\lceil L(w)/T_b \rceil} = 1 - (1-p)^{\lceil w/CT_b \rceil} \tag{7.4}$$

TCP can also transmit packets not in a back-to-back fashion if Pacing [42] is adopted. Pacing tries to evenly spread packets in one window to one RTT slot. Figure 7.3 shows a typical packet pattern of Pacing. It can be seen that packets in one RTT will appear in more optical bursts in the Pacing scheme. Therefore, the TCP packet drop probability increases, the TCP window size drops more frequently, and the throughput is expected to become lower in the Pacing scheme. If Pacing is adopted, the number of optical bursts that contains the packets in one RTT is:

$$N(w) = \min\left\{w, \frac{RTT}{T_b}\right\} \tag{7.5}$$

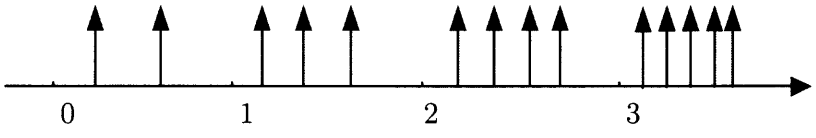

Figure 7.3: Illustration of Packet Pattern with Pacing.

Therefore, the probability that TCP packets get lost due to the optical burst loss in one RTT is:

$$q(w) = 1 - (1-p)^{N(w)} \tag{7.6}$$

Note that Eq. 7.6 is very different from Eq. 7.4, which is the case when TCP Pacing is not implemented. The performance comparison will be presented later in this section. Since the focus is on the best throughput HSTCP can achieve, the time to retransmit lost packets is neglected, and the schemes to detect false Time Out [41] are assumed to be implemented.

Theorem 7.1 *The upper bound of HSTCP throughput is:* $\frac{1}{RTT}\left(1 - \frac{b(w_0^*)}{2}\right) w_0^*$, *where* w_0^* *satifies*

$$\frac{b(w_0^*)w_0^*}{a(w_0^*)} = \frac{1}{q((1-b(w_0^*))w_0^*)}. \tag{7.7}$$

Here, $q(w)$ *is expressed in Eq. 7.4.*

The proof of Theorem 7.1 is given at the end of this chapter. For the throughput upper bound of HSTCP Pacing, the similar result can be found:

Corollary 7.1 *The throughput upper bound of HSTCP Pacing is:* $\frac{1}{RTT}\left(1-\frac{b(w_0^*)}{2}\right)w_0^*$, *where* w_0^* *satifies*

$$\frac{b(w_0^*)w_0^*}{a(w_0^*)} = \frac{1}{q((1-b(w_0^*))w_0^*)}.$$

Here, $q(w)$ *is expressed in Eq. 7.6.*

The proof of Corollary 7.1 is very similar to that of Theorem 1, and it is thus omitted.

Figure 7.4 shows the throughput of HSTCP without Pacing for different values of optical burst loss ratio p and the burst assembly time T_b with the help of Theorem 7.1. Here, the link capacity C is set to 10 Gbps, the packet size to 1500 bytes, RTT to 100ms, and the HSTCP throughput without optical burst loss to one. Note that the optical burst loss ratio p and burst assembly time T_b have great impact on the throughput of HSTCP. With the same value of the burst loss ratio, the smaller the assembly time T_b is, the lower the throughput of HSTCP is.

Fig. 7.5 shows the performance comparison among TCP (without Pacing), HSTCP Pacing, and HSTCP over the OBS network. For TCP, $a(w) = 1$ and $b(w) = 0.5$ according to the default values for the standard TCP. The link capacity C is 10 Gbps, the burst assembly time T_b is 1ms, RTT is 100ms, and the HSTCP throughput without optical burst loss is set to one. It can be seen that although the optical burst loss can greatly reduce the throughput of HSTCP, HSTCP has much higher throughput than TCP. This is because HSTCP increases its window size more aggressively and decreases it more conservatively. One important feature shown in the figure is the throughput of HSTCP Pacing. Though HSTCP Pacing adopts the same aggressive window increasing and conservative window decreasing schemes as HSTCP does, its throughput is significantly smaller than that of HSTCP. It is because the Pacing scheme tends to spread TCP packets evenly, thus leading to much higher packet dropping probability. Furthermore, the throughout of HSTCP can

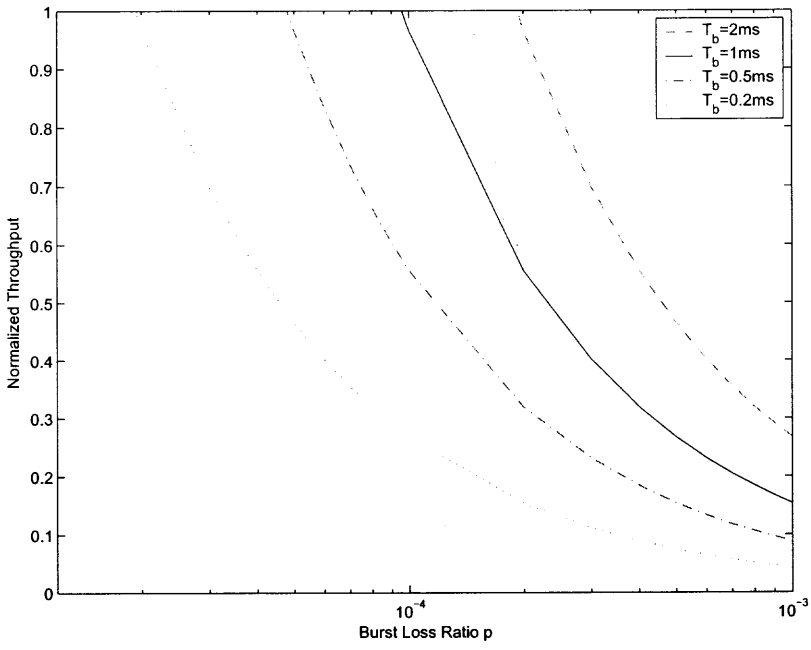

Figure 7.4: Throughput Performance of HSTCP.

be considerably smaller than that of TCP when the optical burst loss ratio is relative large; when the optical burst loss ratio increases, the throughput of HSTCP Pacing will catch up and finally exceed that of TCP. This demonstrates that it should be very cautious in adopting the Pacing scheme if the backbone network is OBS based.

7.3 Summary

In this chapter, a simple model for a single HSTCP connection over an OBS network is presented. Based on this model, the impact of OBS on the throughput of a single HSTCP connection is investigated. The analysis shows that the high burst loss ratio and the small burst assembly time can significantly reduce the throughput of the HSTCP connection. The results in this chapter indicate that improper design of the OBS core network may limit the throughput of HSTCP.

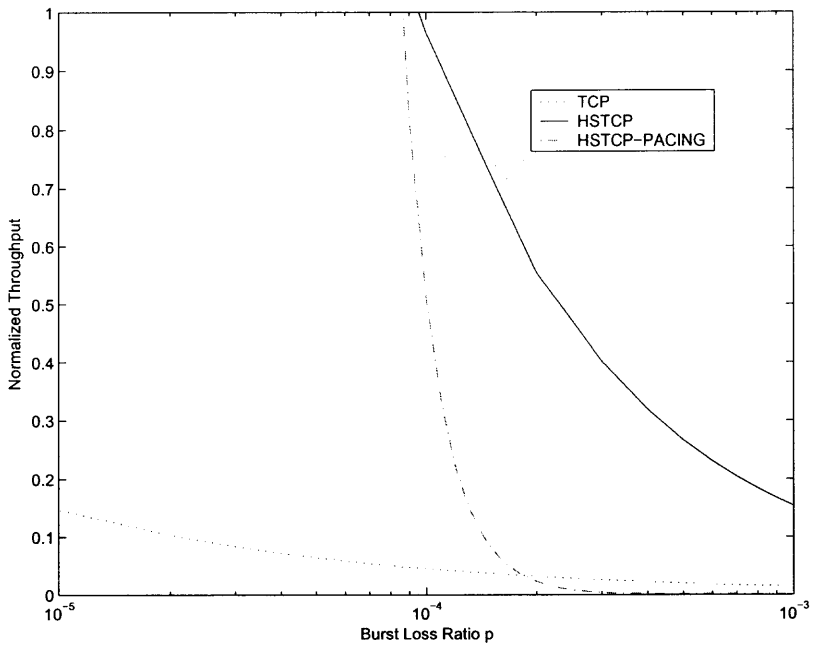

Figure 7.5: Throughput Comparison among TCP, HSTCP, and HSTCP Pacing.

7.4 Proof of Theorem 7.1

The deterministic model similar to [43] is adopted. In the deterministic model, the window size evolves like the saw tooth shape, which is illustrated in Figure 7.6. When the HSTCP window size reaches its maximum w^*, an optical burst loss happens, and the window size drops to $(1 - b(w^*))w^*$. Then, the window keeps increasing to w^*, and the optical burst loss happens again, which leads to another window dropping. This process repeats continuously.

The average throughput of HSTCP can be expressed as:

$$R = \frac{1}{t_1 - t_0} \int_{t_0}^{t_1} \frac{w(t)}{RTT} dt. \tag{7.8}$$

Between t_0 and t_1, the window size evolves according to

$$\frac{dw}{dt} = \frac{a(w)}{RTT}. \tag{7.9}$$

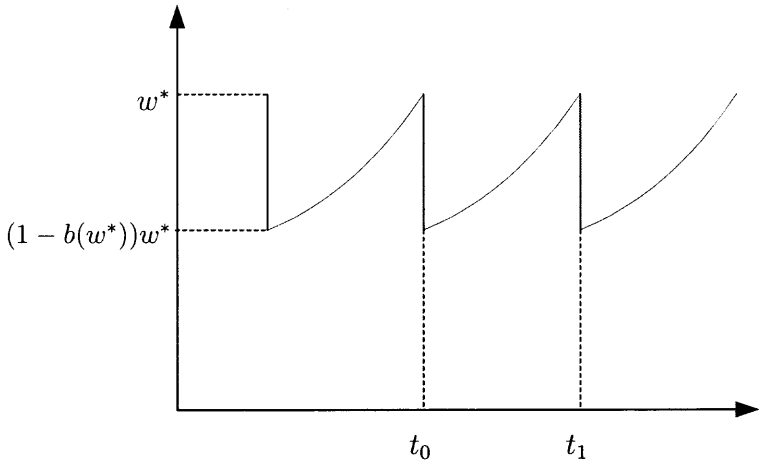

Figure 7.6: HSTCP Window Evolution in Deterministic Model.

Based on the fact that $a(w)$ is an increasing function of w, it is known that $\frac{dw}{dt}$ is an increasing function of time t from Eq. 7.9. Thus, $w(t)$ is a convex function of time t between t_0 and t_1 [44]. Using the property of convex function,

$$w(t) \leq w(t_0) + \frac{w(t_1) - w(t_0)}{t_1 - t_0} \times t \qquad t_0 \leq t \leq t_1 \tag{7.10}$$

Substituting Eq. 7.10 into Eq. 7.8,

$$\begin{aligned} R &= \frac{1}{t_1 - t_0} \int_{t_0}^{t_1} \frac{w(t)}{RTT} dt \\ &\leq \frac{1}{(t_1 - t_0)RTT} \int_{t_0}^{t_1} \left\{ w(t_0) + \frac{[w(t_1) - w(t_0)] \times (t - t_0)}{t_1 - t_0} \right\} dt \\ &= \frac{1}{(t_1 - t_0)RTT} \int_{t_0}^{t_1} \left\{ (1 - b(w^*))w^* + \frac{b(w^*)w^* \times (t - t_0)}{t_1 - t_0} \right\} dt \\ &= \frac{1}{RTT} \left\{ 1 - \frac{b(w^*)}{2} \right\} w^* \end{aligned} \tag{7.11}$$

Assume that

$$t_1 - t_0 = M \times RTT, \tag{7.12}$$

where M is an positive integer. It is known from Eq. 7.4 that when the window size is w, the probability that TCP packets get lost due to the optical burst loss in one RTT is $q(w)$. If $q(w) = q$, which means it is independent of w,

$$M = \frac{1}{q}, \tag{7.13}$$

However, $q(w)$ is not a constant as w increases from $(1 - b(w^*))w^*$ to w^*. Based on the fact that $q(w)$ is an increasing function of w,

$$q(w^*) \leq q(w) \leq q((1 - b(w^*))w^*), \text{ for } (1 - b(w^*))w^* \leq w \leq w^* \tag{7.14}$$

From Eq. 7.13 and 7.14,

$$\frac{1}{q(w^*)} \leq M \leq \frac{1}{q((1 - b(w^*))w^*)} \tag{7.15}$$

On the other hand, from Eq. 7.9,

$$\int_{t_0}^{t_1} \frac{a(w)}{RTT} dt = \int_{t_0}^{t_1} \frac{dw}{dt} dt = w(t_1) - w(t_0) = b(w^*)w^* \tag{7.16}$$

From Eq. 7.9 and the property that $a(w)$ is an increasing function of w,

$$\int_{t_0}^{t_1} \frac{a((1 - b(w^*))w^*)}{RTT} dt \leq \int_{t_0}^{t_1} \frac{dw}{dt} dt \leq \int_{t_0}^{t_1} \frac{a(w^*)}{RTT} dt \tag{7.17}$$

Simplifying Eq. 7.17

$$\int_{t_0}^{t_1} \frac{a((1 - b(w^*))w^*)}{RTT} dt \leq \int_{t_0}^{t_1} \frac{dw}{dt} dt \leq Ma((1 - b(w^*))w^*) \leq \int_{t_0}^{t_1} \frac{dw}{dt} dt \leq Ma(w^*). \tag{7.18}$$

Combining Eq. 7.16 and 7.18,

$$\frac{b(w^*)w^*}{a(w^*)} \leq M \leq \frac{b(w^*)w^*}{a((1 - b(w^*))w^*)}. \tag{7.19}$$

Equations 7.17 and 7.19 set the range of the (w^*, M) pair, which is the shaded area indicated in Figure 7.7. It can be verified that $\frac{1}{q(w^*)}$ and $\frac{1}{q((1-b(w^*))w^*)}$ are both decreasing

functions of w^*, and $\frac{b(w^*)w^*}{a(w^*)}$ and $\frac{b(w^*)w^*}{a((1-b(w^*))w^*)}$ are both increasing functions of w^*. Thus, w_0^*, the upper bound of w^* is achieved at point A, and it satisfies

$$\frac{b(w_0^*)w_0^*}{a(w_0^*)} = \frac{1}{q((1-b(w_0^*))w_0^*)}. \tag{7.20}$$

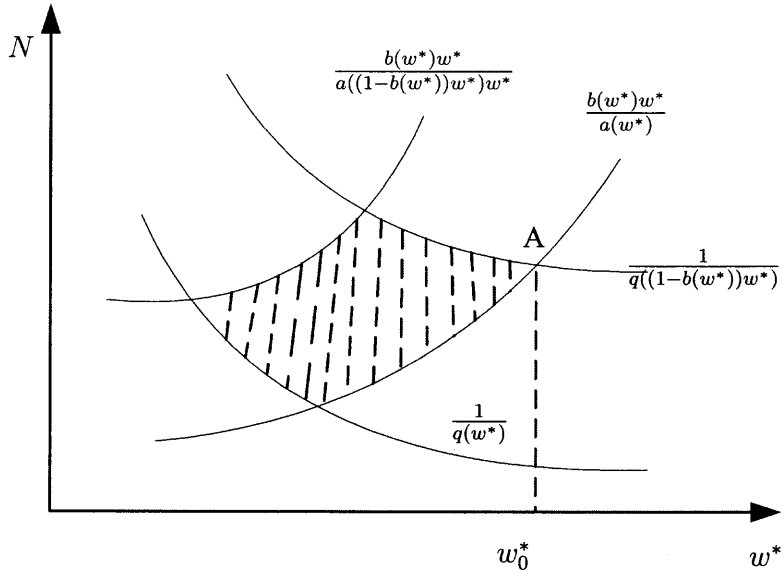

Figure 7.7: Throughput Comparison among TCP, HSTCP, and HSTCP Pacing.

From the property of $b(w)$, it is known that $\left(1-\frac{b(w^*)}{2}\right)w^*$ is an increasing function of w^*. Thus, from Eq. 7.11 and 7.20, the upper bound of the throughput of HSTCP can be expressed as:

$$R \leq \frac{1}{RTT}\left\{1-\frac{b(w_0^*)}{2}\right\}w_0^*. \tag{7.21}$$

CHAPTER 8

SUMMARY AND FUTURE WORK

This chapter concludes this dissertation by summarizing the finished works and discussing the possible directions for the future work.

8.1 Summary

Chapter 2 first identifies the end-to-end delay divergence problem faced by EF PHB. Then, it presents the Youngest Serve First (YSF) aggregate scheduling scheme for the EF class and the corresponding end-to-end delay bound of this scheme. YSF provides a much smaller end-to-end delay bound than that of FIFO and solves the end-to-end delay divergence problem faced by FIFO. YSF does not need to maintain per flow information and its scheduling complexity is independent of the number of flows traversing the link. Therefore, YSF is efficient, effective and scalable, and it is a very promising candidate for scheduling the EF traffic.

Chapter 3 first introduces the necessary background on REM and the TCP model. Then, the local stability condition for REM is presented in the multi-link and multi-source network with an arbitrary uniform feedback delay under the continuous time model. These results provide valuable insight into how to set REM parameters so that a locally stable network can be obtained

Chapter 4 proposes the Virtual Queue and Rate (VQR) based AQM scheme and studies the local stability property for a network with arbitrary topology and delays. VQR is targeted to enhance the currently deployed TCP protocols, and does not require changes made to end users in order to maintain network local stability.

Chapter 5 first identifies the drawbacks of traditional AQM schemes. Then, the edge based AQM (EAQM) is proposed. Compared with conventional AQM schemes, which

need to be deployed over the entire network, EAQM is a pure edge based approach, in which only the edge routers are upgraded. It is easier and more economical to deploy EAQM than the traditional AQM schemes. Although EAQM is simpler than the conventional AQM schemes, EAQM achieves similar or better performance in terms of queue length stability and responsiveness to traffic dynamics. Moreover, EAQM can significantly reduce the throughput unfairness against TCP connections with longer RTTs.

Chapter 6 investigates the fair bandwidth allocation problem in the context of the AF service. It first extends the fairness criteria from single bottleneck link AF networks to those with an arbitrary number of bottleneck links. Then, a new network-assist packet marker (NPM) is proposed to realize the extended fairness criteria and is proven to guarantee the new fairness criteria via theoretical analysis and extensive experiments.

Chapter 7 presents a simple model for a single HSTCP connection over an OBS network. The analysis shows that the high burst loss ratio and the small burst assembly time can significantly reduce the throughput of the HSTCP connection. The results in this chapter indicate that improper design of the OBS core network may limit the throughput of HSTCP.

8.2 Future Work

There are several important directions in which future research can be pursued:

1. *QoS in wireless networks* — Providing QoS in the wireless networks requires different approaches from those in the traditional wired network because the link capacity, queuing delay and packet loss are much more difficult to control in the wireless networks. One interesting and challenging topic is to provide service differentiation and to optimize the network performance in both pure wireless networks and the hybrid networks. The approach is a cross layer design, which takes multiple factors into consideration, i.e., power control, flow and congestion control, packet scheduling, user mobility and routing.

2. *Overlay network* — One of the main purposes of overlay networks is to provide end-to-end QoS on top of the best effort Internet infrastructure. How to extend the work in this dissertation to overlay network is a new research opportunity. Since each overlay network aims to provide QoS in a smaller dimension than the DiffServ networks, there will be freedom to design algorithms to realize service differentiation based on the particular requirement of each application with less complexity and scalability constraints.

REFERENCES

[1] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: An Overview," IETF RFC 1633, 1994.

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services ," IETF RFC 2475, 1998.

[3] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP) - Version 1, Functional Specification," IETF RFC 2205, 1997.

[4] S. Shenker, C. Partridge, R. Guerin, "Specification of Guaranteed Quality of Service," IETF RFC 2212, 1997.

[5] J. Wroclawski, "Specification of the Controlled-Load Network Element Service," IETF RFC 2211, 1997.

[6] A. Parekh and R. Gallagher, "A Generalized Processor Sharing Approach to Flow Control - the Single Node Case," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 366–57, June 1993.

[7] V. Jacobson, "Congestion Avoidance and Control," *Proceedings of ACM SIGCOMM 1988*, pp. 314–329, August 1988.

[8] S. Ryu, "Advances in Internet Congestion Control," *IEEE Communication Surveys and Tutorials*, vol. 5, no. 1, pp. 28–39, Third Quarter 2003.

[9] K. Thompson, G. J. Miller, and R. Wilder, "Wide-Area Internet Traffic Patterns and Characteristics", *IEEE Network*, vol. 11, no. 6, pp. 10–23, November/December 1997.

[10] A. Charny and J.-Y. Le Boudec, "Delay Bounds in a Network with Aggregated Scheduling," *Proceedings of QoSFIS*, pp. 1–13, October 2000.

[11] J. Bennet, K. Benson, A. Charny, W. Courtney, and J. Y. Le Boudec, "Delay Jitter Bounds and Packet Scale Rate Guarantee for Expedited Forwarding," *Proceedings of IEEE INFOCOM, 2001* vol. 3, pp. 1502–1509, April 2001.

[12] J. Y. Le Boudec and P. Thiran, *Network Calculus, A Theory of Deterministic Queuing Systems for the Internet*, Springer Verlag - LNCS 2050, June 2000.

[13] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, pp. 397–413, August 1993.

[14] K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," IETF RFC 3168, 2001.

[15] C. Hollot, V. Misra, D. Towsley, and W. Gong, "On Designing Improved Controllers for AQM Routers Supporting TCP Flows," *Proceedings of IEEE INFOCOM 2001*, vol. 3, pp. 1726–1734, April 2001.

[16] S. Kunniyur and R. Srikant, "Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management," *Proceedings of ACM SIGCOMM 2001*, pp. 123–134, August 2001.

[17] S. Athuraliya, S. H. Low, V. H. Li, and Q. Yin, "REM: Active Queue Management," *IEEE Network Magazine*, vol. 15, no. 3, pp. 48–53, May/June 2001.

[18] Y. Gao and J. C. Hou, "A State Feedback Control Approach to Stabilizing Queues For ECN-Enabled TCP Connections" *Proceedings of IEEE INFOCOM 2003*, vol. 3, pp. 2301–2311, April 2003.

[19] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," IETF RFC 2001, 1997.

[20] S. Floyd, T. Henderson, and A. Gurtov, "The NewReno Modification to TCP's Fast Recovery Algorithm," IETF RFC 3782, 2004.

[21] S. Floyd, J. Hadi Salim, and U. Ahmed, "An Extension to the Selective Acknowledgment (SACK) Option for TCP," IETF RFC 2884, 2000.

[22] S. Floyd, "HighSpeed TCP for Large Congestion Windows," IETF RFC 3649, Experimental, 2003.

[23] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 5, pp. 1465–1480, October 1995.

[24] C. Jin, D. X. Wei, and S. H. Low, "FAST TCP: Motivation, Architecture, Algorithms, Performance," *Proceedings of IEEE INFOCOM 2004*, vol. 4, pp. 2490–2501, March 2004.

[25] W. Feng, "A Self-Configuring RED Gateway," *Proceedings of IEEE INFOCOM 1999*, vol. 3, pp. 1320–1328, March 1999.

[26] T. J. Ott, T. V. Lakshman, and L. Wong, "SRED: Stabilized RED," *Proceedings of IEEE INFOCOM 1999*, vo. 3, pp. 1346–1355, March 1999.

[27] P. Ranjan, E. H. Abed, and R. J. La, "Nonlinear Instabilities in TCP-RED," *Proceedings of IEEE INFOCOM 2002*, vol. 12, pp. 248–258, June 2002.

[28] S. H. Low, F. Paganini, J. Wang, S. Adlakha, and J. C. Doyle, "Dynamics of TCP/RED and a Scalable Control," *Proceedings of IEEE INFOCOM*, vol. 1, pp. 239–248, June 2002.

[29] F. Paganini, "On the Stability of Optimization Based Flow Control," *Proceedings of American Control Conference*, June 2001.

[30] Q. Yin and S. H. Low, "Convergence of REM Flow Control at a Single Link," *IEEE Communications Letters*, vol. 5, no. 3, pp. 119–121, March 2001.

[31] Q. Yin and S. H. Low, "On Stability of REM Algorithm with Uniform Delay," *IEEE GLOBECOM 2002*, pp. 2649–2653, November 2002.

[32] S. H. Low, "A Duality Model of TCP Flow Control," *Proceedings of ITC Specialist Seminar on IP Traffic Management, Modeling and Management*, September 2000.

[33] S. H. Low, L. Peterson, and L. Wang, "Understanding Vegas: a Duality Model," *Journal of ACM*, vol. 49, no. 2, pp. 207–235, March 2002.

[34] B. Noble and J. W. Daniel, *Applied Linear Algebra*, 3rd edition, Prentice Hall, New Jersey, 1988.

[35] D. Katabi, M. Handley, and C. Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks," *ACM Proceedings of SIGCOMM 2002*, pp. 89–102, August 2002.

[36] 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, no. 5, pp. 633–647, December 2003.

[37] G. Vinnicombe, "On the Scalability of End-to-End Congestion Control for the Internet," Technical report, Cambridge University, CUED/F-INFENG/TR.398, December 2000.

[38] S. Floyd, "HighSpeed TCP for Large Congestion Windows," IETF RFC 3649, Experimental, 2003.

[39] J. Liu, N. Ansari, and T. Ott, "FRR for Latency Reduction and QoS Provisioning in OBS Networks," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 7, pp. 1210–1219, September 2003.

[40] A. Detti, M. Listanti, "Impact of Segments Aggregation of TCP Reno Flows in Optical Burst Switching Networks," *Proceedings of IEEE INFOCOM 2002*, vol. 3, pp. 1803–1812, June 2002.

[41] X. Yu, C. Qiao, and Y. Liu, "TCP Implementations and False Time Out Detection in OBS Networks," *Proceedings of IEEE NFOCOM 2004*, vol. 2, pp. 774–784, March 2004.

[42] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the Performance of TCP Pacing," *Proceedings of IEEE INFOCOM 2000*, vol. 3, pp. 1157–1165, March 2000.

[43] S. Sahu, P. Nain, D. Towsley, C. Diot, and V. Firiou, "On Achievable Service Differentiation with Token Bucket Marking for TCP," *Proceedings of ACM SIGMETRICS 2000*, vol. 28, no. 1, pp. 23–33, June 2000.

[44] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

[45] L. Zhu and N. Ansari, "Local Stability of a New Adaptive Queue Management (AQM) Scheme," *IEEE Communications Letters*, vol. 8, no. 6, pp 406–408, June 2004.

[46] C. Albuquerque, B. J. Vickers, and T. Suda, "Network Border Patrol: Preventing Congestion Collapse and Promoting Fairness in the Internet," *IEEE/ACM Transactions on Networking*, vol. 1, no. 1, pp. 173–186, January 2004.

[47] Y. C. Chan, C. T. Chan, and Y. C. Chen, "An Enhanced Congestion Avoidance Mechanism for TCP Vegas," *IEEE Communications. Letters*, vol. 7, no. 7, pp. 343–345, July 2003.

[48] C. C. Hollot, V. Misra, D. Towsley, and W. B. Gong, "A Control Theoretical Analysis of RED," *Proceedings of IEEE INFOCOM 2001*, vol. 3, pp. 1510–1519, April 2001.

[49] D. Clark and W. Fang, "Explicit Allocation of Best-Effort Packet Delivery Service," *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, pp. 362–373, August 1998.

[50] W. Feng, D. Kandlur, D.Saha, and K. Shin, "Adaptive Packet Marking for Maintaining End-to-End Throughput in a Differentiated-Services Internet," *IEEE/ACM Transactions on Networking*, vol. 7, no. 5, pp. 685–697, October 1999.

[51] B. Nandy, N. Seddigh, P. Pieda, and J. Ethridge, "Intelligent Traffic Conditioner for Assured Forwarding Based Differentiated Services Networks," *Proceedings of High Performance Networking 2000 Conference*, 2000.

[52] I. Yeom and A. L. Reddy, "Modeling TCP Behavior in a Differentiated Services Network," *IEEE/ACM Transactions on Networking*, vol. 9, no. 1, pp. 31–46, February 2001.

[53] M. A. El-Gendy and K. Shin, "Equation-based Packet Marking for Assured Forwarding Based Differentiated Services Networks," *Proceedings of IEEE INFOCOM 2002*, vol. 2, pp. 845–854, March 2002.

[54] B. Nandy, J. Ethridge, A. Lakeas, A. Chapman, "Aggregate Flow Control: Improving Assurances for Differentiated Services Network," *Proceedings of IEEE INFOCOM 2001*, vol. 3, pp. 1340–1349, April 2001.

[55] Y. Chait, C. Hollot, V. Misra, D. Towsley, and H. Zhang, "Providing Throughput Differentiation for TCP Flows Using Adaptive Two Color Marking and Multi-level AQM," *Proceedings of IEEE INFOCOM 2001*, vol. 2, pp. 837–844, April 2001.

[56] S. Herreia-Alonso, A. Suarez-Gonzalez, M. Fernandez-Veiga, R. F. Rodriguez-Rubio, C. Lopez-Garcia, "Improving Aggregate Flow Control in Differentiated Services Networks," *Computer Networks*, vol. 44, pp. 499–512, March 2004.

[57] E. Park ad C. Choi, "Proportional Bandwidth Allocation in DiffServ Networks," *Proceedings of IEEE INFOCOM 2004*, vol. 3, pp. 2038–2049, March 2004.

[58] N. Seddigh, B. Nandy, and P. Pieda, "Bandwidth Assurance Issues for TCP Flows in a Differentiated Services Network," *Proceedings of IEEE Globecom 1999*, vol. 3, pp. 1792–1798, March 1999.

[59] J. -A. Ibanez, K. Nicholas, "Preliminary Simulation Evaluation of an Assured Service," IETF Draft, August 1998.

[60] A. Chapman and H. T. Hung, "Traffic Management for Aggregate IP Streams," *Proceedings of CCBR*, Ottawa, November 1999.

[61] D. Bertsekas, R. Galager, *Data Networks*, Ch. 6, Prentice Hall, Englwood Cliffs, 1992.

[62] A. Arulambalam, X. Chen and N. Ansari, "Allocating Fair Rates for Available Bit Rate Services in ATM Networks," *IEEE Communications Magazine*, vol. 34, no. 11, pp. 92–100, November 1996.

[63] A. Charny, D. D. Clark, and R. Jain, "Congestion Control with Explicit Rate Indication," *Proceedings of IEEE INFOCOM 1995*, vol. 3, pp. 1954–1963, April 1995.