WASHINGTON UNIVERSITY

SCHOOL OF ENGINEERING AND APPLIED SCIENCE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CONGESTION MANAGEMENT FOR LOSSLESS ETHERNET OPERATION

by

Jinjing Jiang

Prepared under the direction of Professor Raj Jain

A thesis presented to the School of Engineering of
Washington University in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

May 2008

Saint Louis, Missouri

WASHINGTON UNIVERSITY

SCHOOL OF ENGINEERING AND APPLIED SCIENCE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ABSTRACT

CONGESTION MANAGEMENT FOR LOSSLESS ETHERNET OPERATION

by

Jinjing Jiang

ADVISOR: Professor Raj Jain

May 2008

Saint Louis, Missouri

IEEE 802.1 standards committee is working on a new specification for congestion notification in Ethernet networks. The goal of this work is to enable applications of Ethernet in backend data center networks. Such applications typically use Fiber Channel and Infiniband due to their loss-free characteristics. In this thesis, an explicit rate control framework called Forward Explicit Rate Advertising (FERA) for Ethernet applications, especially data centers, is described. The framework guarantees zero packet drops at the congested switches and fast convergence to fair and stable state. In order to manage the congestion, design choices on 2-point and 3-point architectures, the reactive and proactive signaling, explicit and implicit rate controls are compared. Then the core component of the framework, queue control, is carefully studied. Furthermore, we show that this framework can seamlessly cooperate with IEEE 802.3x PAUSE mechanism to recover from severe congestion scenarios. To demonstrate the strength of FERA, both analytic and simulation results comparing with another existing scheme called Backward Congestion Notification (BCN) are provided to show that FERA achieves the design goals.

Contents

List of Tables

List of Figures

Acknowledgments

I would like to thank first my advisor, Professor Raj Jain, for bringing me to the research area of congestion control, and guiding me in the past three years. I am deeply grateful to him for his availability and help, not only in research, but also in all other aspects of the life. I am also very grateful to him for taking so much care of his student's personal career and promoting their results.

I would like to extend my sincere thanks to Dr. John Z. Yu for the time he spent assisting me with my research, from pointing me to important literatures to advising me to attack a problem in the systematic manner.

Lastly, I would like to extend my greatest gratitude towards my family. Their endless love is the only origin of all my courage.

Jinjing Jiang

Washington University in Saint Louis
May 2008

Chapter 1

Introduction

Recently, due to the storm of data growth inside sizable enterprise, super servers, clusters, bladed systems are interconnected where huge amounts of data are exchanged or stored all the time. The IEEE 802.1 standards committee has been discussing the possibility of using Ethernet as the infrastructure to enable the data center applications. For simplicity, in the following we call this kind of network as Data Center Ethernets (DCEs). Furthermore, the necessity of traffic engineering in DCEs has been discussed for over a year now and has approved a project authorization request to develop a standard for congestion management. This specification when completed will be known as IEEE 802.1Qau.

Generally speaking, DCEs are one kind of high speed network of limited network diameter. The typical aggregate link throughput ranges from 10 Gbps to 100 Gbps. On the other hand, DCEs require low latency for efficient communications. Due to the large number of end stations configured in clusters, these networks are vulnerable to link congestion. For example, in a typical star topology running TCP traffic, the output link of the core switch could be congested frequently resulting in packet losses and timeouts that can severely jeopardize the overall system throughput due to the retransmission policy. This will lead to intolerable long latency for successful transmissions. Empirical data shows that switch buffers can overflow resulting in packet loss even when the average utilization is 50% due to bursty nature of the traffic.

While significant amount of work has been done on congestion control in TCP/IP networks, Ethernet networks, even after 30 years of their invention, run without congestion control in the data link layer. This may be acceptable for elastic applications but not tolerable for data center applications. The packet loss rate in data center applications should be practically zero. This is why traditionally data centers have used Fiber Channel and Infiniband networks that provide hop-by-hop flow control and sophisticated congestion control mechanisms to avoid packet losses.

Data centers cannot rely solely on TCP to take care of network congestion. There are many applications that use UDP and some don't even use IP, e.g., Veritas Cluster os. It is, therefore, important that Ethernet networks provide a congestion control mechanism in the data link layer.

In this thesis, we mainly concentrate on the Forward Explicit Rate Advising mechanism (FERA) we proposed and the comparison to another existing proposal called Backward Congestion Notification (BCN). The main contributions of this thesis is summarized as the following.

- An explicit rate based congestion control mechanism is proposed for Ethernet data center applications. To the best knowledge of the author, this is the first working proposal using forward path probing (rate discovery) methodology in Ethernet.
- Both analytical and numerical results are provided to show that FERA outperforms BCN in various metrics, especially in the convergence time and fairness.

The thesis is organized as follows. Chapter 2 reviews the related work in the literature. In Chapter 3, the design goals and choices for mechanism design are explained. In Chapter 4 and 5, the detailed system model for FERA mechanism is described. In Chapter 6, analytical results of convergence rate on both FERA and BCN are presented. Chapter 7 gives various enhancement on FERA in practical implementation. The simulation results that support our claims are provided in Chapter 8. Then the thesis ends with the conclusions.

Chapter 2

Related Work

In the literature, McAlpine and Wadekar [1] proposed the general architecture for congestion management in Ethernet Clusters, where link level control, layer 2 subnet control and end-to-end higher layer control protocols are discussed. Through simulations, they endeavor to find the appropriate set of congestion management methods that are compatible with IEEE802.1/802.3 standards. Santos et al[2] described a simple switch-based ECN mechanism for Infiniband with a new source rate control mechanism using window limit. It improves fairness and throughput for both static and dynamic traffic. However, this scheme only works with TCP traffic.

In the IEEE 802.1 standards group, four proposals are currently being discussed. First, a Backward Congestion Notification (BCN) was proposed by Davide Bergamasco and his colleagues at Cisco and is described in [3]. The BCN control messages are sent from switches back to sampled sources when the congestion happens. The feedback contains current queue status including queue length and variation. Then the sources basically adopts the Additive Increase and Multiplicative Decrease (AIMD) mechanism used in TCP to adjust the sending rate[4][5]. We proposed a Forward Explicit Rate Advertising mechanism (FERA) and argued in favor of explicit feedback of allowed rates to the sources. FERA uses the similar idea proposed in Rate Control Protocol(RCP)[6]. However, FERA does not have per packet acknowledgement and the queue control is quite different. A modification to BCN, called Explicit Ethernet Congestion Management (E2CM) by IBM [7] was proposed as combining some of the ideas of FERA and BCN. The fourth proposal is quantized congestion notification (QCN) in which BCN queue feedback is quantized to a few bits and network provides only negative feedback [7]. Generally, QCN is a simplified version of BCN. There are no positive feedback messages allowing sources to increase the rate. Instead, sources use a search algorithm to find the acceptable rate, which is very similar to the BIC/CUBIC proposed in [8].

Chapter 3

Design Goals and Choices

3.1 Performance Goals

In designing a working scheme for data center applications, the following metrics are of common interest.

- *High Throughput and Low Queueing Delays*: Since the demand for data exchange and storage in DCEs is extremely large compared with other networks, the goodput of the network should be maximized. In other words, the aggregate link throughput should also be maximized. Furthermore, in order to keep the delays under some acceptable value even with high link utilization, we aim to control the queue length at a constant level. Thus, the variation of the latency is minimized.
- *Stability, Convergence Rate and Fairness*: The stability of a system is the common control target. Even for some very bursty traffic, the scheme possibly has large oscillations. However, as long as the stochastic average of the queue length is be bounded around the target level, we regard the system stable. Furthermore, the time to converge to stable state is also an important metric of system performance. For example, for bursty traffic, if the convergence time is longer than the average busy period, the mechanism definitely cannot handle the traffic dynamics properly. Also from the user side of view, some form of the fairness should be guaranteed. Therefore each flows going through one bottleneck must achieve some form of weighted fairness [9], for example, max-min or proportional fairness.

- *Zero Loss at the Bottleneck*: In order to eliminate the effect of retransmission, zero loss at the congested switch is enforced. However, the traffic load in the network is unpredictable, so it is almost impossible to ensure zero loss only by a feedback mechanism. So the PAUSE mechanism used in IEEE 802.3x is introduced to adopt the hop by hop link control by simply asking the upstream neighbor to stop dequeuing packets when the queue length at the bottleneck is already larger than some threshold. In this way, no packets will be lost.

- *Easy Deployment*: If a scheme has many sophisticated parameters or the parameters are hard to tune, the system administrators will find it difficult to manage. Compared with other competing proposals, FERA has smaller number of parameters. It is easy to find the admissible values for these parameters. Furthermore, these values are applicable to a wide range of network configurations, such as link capacity, traffic types and number of sources.

In this paper, we provide both analytical and simulation results to show FERA works better than BCN to achieve these goals.

3.2 Design Choices for Congestion Management

In the IEEE802.1 standards group, architectures using either backward notification, forward probing or both are under debate. This results in two types of architectures as discussed below.

3.2.1 2-Point and 3-Point Architectures

In our proposed FERA scheme, sources periodically generate probing packets that are modified by the switches along the path and then the probing packets are reflected by the destinations back to the sources. The sources react to the feedback received in the returning probes and set their rate accordingly. Thus, there are three types of points in the control loop: reaction points at the sources, congestion points at the switches, and reflection points at the destination. This is known as the 3-point architecture.

It is also possible to have a control loop without reflection points. In this case, the sources react to the feedback received directly from the congestion point. The feedback is sent in the backward direction from the switch to the source. This is the 2-point architecture as shown in Figure 3.1.

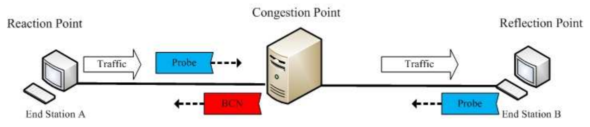


Figure 3.1: The architecture for congestion notification.

BCN uses 2-point architecture. If there are multiple congestion points on the path of a flow, multiple backward control messages will be sent back while only one of these - one with the highest level of congestion indication - will dominate the future rate of the flow. Sophisticated mechanisms are needed for sources to respond to messages from different congestion points. While in FERA, the sources do not have to keep track of congestion points and can increase their rate as indicated the received feedback immediately.

3.2.2 Proactive and Reactive Signaling

In some scenarios, a sudden change of the link capacity or traffic patterns will cause the network to be severely congested. Since proactive probes are sent only periodically, at least one periodic interval is needed to respond to the sudden overload. This may cause long queues in the switch buffer. In this case, reactive signaling with feedback as soon as the congestion happens will help. In Fig. 3.1, BCN messages are sent from the switch to sources as long as the queue length is above some predefined level. The problem with this approach is that it reduces the rate of some flows too much and then these flows may not recover by the random sampling method used at the congested switch.

3.2.3 Explicit and Implicit Rate Control

BCN, QCN, E2CM, all three proposals send the queue dynamics back to the sources. Then the sources perform the Additive Increase Multiplicative Decrease (AIMD) algorithm to adjust their transmission rate. However, the queue based congestion sensor cannot not tell directly what bandwidth the congested link can support since queue length depends upon the queue service architecture, and is highly related to the bottleneck link rate. For example, ten 1500B packets at a 1 kbps link are a *big* queue while the same queue would be considered negligible at a 10 Gbps link. Therefore, queue length feedback from different links cannot be compared.

On the contrary, FERA uses rate based sensor - the link utilization - to detect the congestion. The switches calculate the fair share for each flow without maintaining any per-flow information. Using the link utilization as congestion indicator is much easier for the network administrator, since desired utilization is same at 1 Gpbs and 10 Gbps links.

Mathematically, queue length is an instantaneous random variable, while rate of sources is a time averaged variable measured in a predefined interval. For the same load, the instantaneous queue length can vary a lot. Consider a simple M/M/1 queue for example. Suppose ρ is the load factor (equivalent to link utilization), the probability that queue length Q is equal to n is

$$P(Q = n) = (1 - \rho)\rho^n.$$

Therefore, rate based sensor is more stable (less variance) than queue-based load sensor.

Furthermore, this explicit rate control messages are much simpler in terms of message format since it is not necessary to indicate the identification of different congestion points. In Chapter 6, we will show that FERA's convergence time to fair and stable state outperforms that of BCN.

In summary, considering both the system performance and complexity issues, the forward explicit rate control is better than implicit rate control schemes.

Chapter 4

FERA Model and Assumptions

4.1 System Model

FERA is a close-loop explicit rate feedback control mechanism as shown in Fig. 4.1. It is assumed that the sources are equipped with the rate regulators, which can be token-bucket traffic shaper. If congestion happens, the switches can effectively measure the current load of the output link, which generally requires several simple counters that can be easily integrated in hardware.

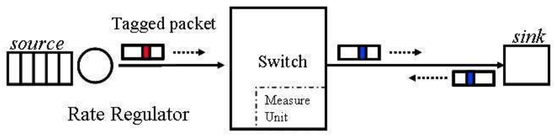


Figure 4.1: Simple FERA system model.

As shown in Fig. 4.1, rate discovery packets are sent periodically in each flow. These packets contain a rate discovery tag. The tags can also be piggybacked on data carrying packets. The measurement unit in the switch monitors the length of its output queue, the input rate and the output capacity. Based on the measurements, the switch periodically calculates the current fair share of its output link and uses it as the advertised rate for sources in the next measurement interval. When the tagged packets go through the switch, the switch reduces the rate in the tag to its advertised rate. When the sink receives these tagged packets, it simply marks it as a returning tag and sends it to the source either as a separate packet or by piggybacking it on the reverse traffic. Of course, the returning tags have a different protocol type to distinguish them from forward tags. At the sources, when a returning tag is received, the rate regulator updates its rate to that indicated in the tag.

Thus the FERA mechanism has 3 components: Measuring, Marking and Reacting. In the following, each component is explained in detail.

4.1.1 Link Measurement

For each link, the network manager sets a target buffer utilization level at equilibrium Q_{eq}. This is the desired number of packets that should be in queue. Another severe congestion threshold Q_{sc} is also set by the network administrator to indicate high congestion level on the link. The switches simply count the number of arriving packets in each measurement interval T. From this count, the switch can calculate the link utilization level in the last measurement interval and use it to set the advertised rate for the next measurement interval. If the congestion is severe, i.e., the queue depth is larger than Q_{sc}, the switch may send a "PAUSE" message on all input ports to prevent loss of packets.

4.1.2 Forward Marking and PAUSE Signaling

FERA has two kinds of signals: PAUSE frames and FERA tags. PAUSE frames are a hop-by-hop congestion control mechanism specified in IEEE802.3x. When the queue length is larger than Q_{sc}, the switch simply sends out PAUSE/ON to all its uplink neighbors. The neighbors stop transmitting packets. This in turn results in neighbor's buffers getting filled up and a PAUSE/ON is issued to the previous hop. Ultimately, PAUSE signal reaches the source end stations. When the queue length in the switch becomes lower than some predefined level, a PAUSE/OFF frame is sent out on the input ports to start transmission of packets. It is well known that PAUSE mechanism can unfairly affect innocent traffic that is not going through the bottleneck and also can reduce the network throughput, so it should be used only rarely and for short time.

4.1.3 Source Tagging and Reaction

The source tags one packet every τ interval. Generally, $\tau \leq T$. The rate r in the tags is initialized to the output link rate by the sources. When the FERA tagged packet reaches the destination, the destination sends the tag back to the source with the new rate r contained in the tag. When the source receives the tag, it updates the rate for the corresponding rate regulator.

Note that FERA works in a defined FERA-aware region of the network. If the packets exit this region, the edge bridges perform the corresponding end system functions and remove the tags from forwarded traffic.

4.2 Assumptions

FERA control messages and tags are specially formatted packets sent through switches in DCEs. It is important that the messages follow a format that is acceptable to legacy switches that are FERA-unaware. Generally, the FERA message format should be VLAN-tagged to ensure the coexistence and interoperability between FERA-aware and FERA-unaware switches. Propagation delay in DCEs is generally small - of the order of a few tens or hundreds of microseconds. This is because the diameter of the network is only a few hundreds meters. Links are assumed to be of high capacity. Most links are either 1 Gigabit per second or 10 Gigabit per second. In our analysis, we assume that switch buffers are FIFO and output queued.

Chapter 5

FERA Switch and Source Algorithms

We now describe the FERA mechanism in detail. Note that each switch allocates the same advertised rate to all flows. It does not need any knowledge of the number of active flows. There is no per-flow accounting in the switches. The switches perform time-based measurement to monitor the load and queue length. Assume $t_0 = 0$, denote the sequential measurement time instants as $t_1, t_2, \ldots, t_i$, where $T = t_i - t_{i-1}$ is the fixed measurement interval. See Fig.5.1. Then, the following variables are

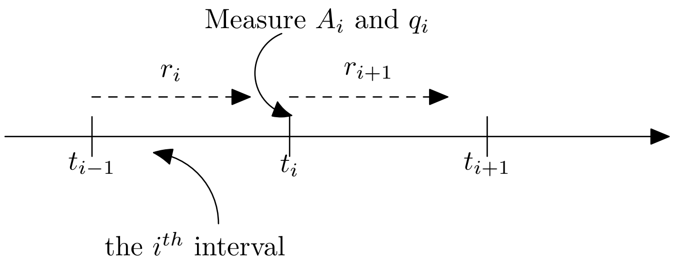


Figure 5.1: Illustration of rate calculation.

defined for the i^{th} interval $(t_{i-1}, t_i]$: ρ_i is the load factor for this interval; q_i is the number of packets in the buffer at the end of the interval (at time t_i); r_i is the advertised rate for the i^{th} interval; A_i is the average arrival rate during the interval. In addition, assume that the link capacity is C, P_m is the packet size. Note that we measure the queue length q_i by the number of packets and use fixed-size packets. However, FERA can handle random packet sizes simply by measuring queue lengths in bytes. In brief, the notation rules are the following: if the value is effective for the

$i+1^{th}$ interval, its subscript is denoted as $i+1$; if the value is measured in the i^{th} interval (at time spot t_i), its subscript is denoted as i.

The basic algorithm is as follows.

- Switch Side:
  - Computation:
    - Initialization: Initial advertised rate $r_0 = \frac{C}{N_0}$; where N_0 is a constant determined by the network administrator.
    - Measurement of effective load: $\rho_i = \frac{A_i}{f(q_i)\times C}$;
    - Bandwidth Allocation: $r_{i+1} = \frac{r_i}{\rho_i}$;
  - Marking: If the $r_{i+1} < r$ where r is the rate value in the rate discovery tags, then $r = r_{i+1}$.
- Source End:
  - The sources start at the initial rate of $r_0 = \frac{C}{N_0}$. Here C is the source link capacity and N_0 is the network wide parameter (same as that used in the switches).
  - Before the source starts its transmission, it sends out a rate discovery tag (either piggybacked on the first data packet or as a separate packet) to find out the bottleneck switch's advertised rate.
  - After the source receives the rate discovery tag, it sets its rate to that in the tag received from the network.
  - If no packets are sent and hence no tags have been received in the last $2T$ time interval, the source resets its rate r to $\frac{C}{N_0}$.

Note that in the last measurement interval, A_i is the sum rate of all input flows at the switch port, thus

$$r_{i+1} = \frac{r_i}{\rho_i} = \frac{Cf(q_i)}{\frac{A_i}{r_i}},$$

where $Cf(q_i)$ could be thought as the *effective* bandwidth available, $N = \frac{A_i}{r_i}$ is the effective number of flows. $f(q)$ is the queue control function [10] to ensure that the queue length is kept at a constant level.

An key feature in the FERA is that we do not need any information about the number of flows, while in ATM congestion control mechanisms, for example the standard algorithm, ERICA [11], the estimation of the number of active flows is required. While in FERA, we have built in the estimated number of flows into the algorithm, which is the key novelty compared with ATM congestion control algorithms. Another difference between FERA and ATM algorithm is that FERA does not need to maintain multiple states as ATM switch does, which leads much simpler operations for Ethernet switches.

5.1 Queue Control Functions

The queue control function $f(q)$ plays a key role in controlling the queue length. In the following discussion, we focus on the linear and hyperbolic functions used in [10]. Generally, if $f(q)$ is a linear function, it has the form

$$f(q) = 1 - k\frac{q - Q_{eq}}{Q_{eq}}, \tag{1}$$

where Q_{eq} is target equilibrium queue length measured by packets, k is some constant. If $f(q)$ is a hyperbolic function,

$$f(q) = \begin{cases} \frac{bQ_{eq}}{(b-1)q+Q_{eq}}, \text{if } q \le Q_{eq}, \\ \max(c, \frac{aQ_{eq}}{(a-1)q+Q_{eq}}), \text{otherwise.} \end{cases} \tag{2}$$

where a, b, c are constants. Note that in both (1) and (2), $f(Q_{eq}) = 1$.

5.2 Analysis of Stability and Fairness

Proposition 1 Without considering the control loop delay, FERA can achieve stability and max-min fairness.

Typically in DCEs, the control loop delay is very small compared with measurement interval. In the following analysis, we ignore the feedback delays.

$$r_{i+1} = \frac{r_i}{\rho_i} = \frac{f(q_i)C}{N}. \tag{3}$$

Then

$$q_{i+1} = q_i + \frac{(Nr_{i+1} - C)T}{P_m} = q_i + \alpha(f(q_i) - 1), \tag{4}$$

where $\alpha = \frac{CT}{P_m}$, P_m is the packet size.

When $f(q)$ is the hyperbolic function, (4) is a nonlinear discrete equation. However, it is easy to show when the initial queue length $q_0 > Q_{eq}$, the sequence $\{q_i\}$ is monotonically decreasing. Similarly, if $q_0 < Q_{eq}$, the sequence $\{q_i\}$ is monotonically increasing. In both cases, the limit of the sequence is exactly Q_{eq} as time goes to infinity, i.e., $\lim_{i\to\infty} q_i = Q_{eq}$. On the other hand, when $q_i = Q_{eq}$, by (3), $f(q_i) = 1$. Therefore the queue length is kept stable at Q_{eq}. Meanwhile, $r_{i+1} = \frac{C}{N}$. It follows that all flows get the fair share, even though we do not know explicitly how many flows there are.

When $f(q)$ is the linear function shown in (1), we can get a closed form solution for q_i, which is written as

$$q_{i+1} = \left(1 - \frac{\alpha k}{Q_{eq}}\right) q_i + \alpha k,$$

then

$$q_i = \left(1 - \frac{\alpha k}{Q_{eq}}\right)^i q_0 + \alpha k \sum_{j=0}^{i} \left(1 - \frac{\alpha k}{Q_{eq}}\right)^j. \tag{5}$$

With a sufficient condition that $\frac{\alpha k}{Q_{eq}} < 1$, $\lim_{i\to\infty} q_i = Q_{eq}$. Note that this sufficient condition is the key criterion to choose the queue control function parameters given the length of measurement interval. With the similar argument for the hyperbolic function, FERA still has the property of convergence to stable state with fair share among each flows. Furthermore, by approximating the hyperbolic function using piecewise linear functions, the above sufficient condition can be used to determine the proper values for a, b and c.

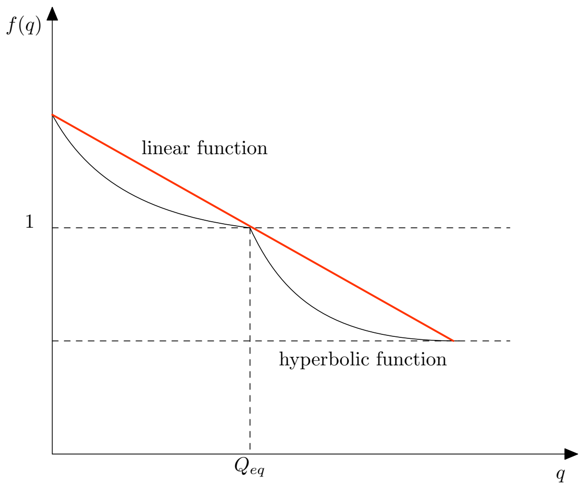


Figure 5.2: The queue control functions.

Chapter 6

Convergence to Fairness

In this chapter, we will provide the analytical results on the convergence time for both FERA and BCN.

6.1 Convergence Time for FERA

Since the hyperbolic queue control function leads to a nonlinear discrete function, it is not trivial to analyze the rate of convergence. However, we can use the linear queue control function to get an approximation (or lower bound) for the rate of convergence. See Fig.5.2. At every point of q, the linear function has a larger value compared with the one of hyperbolic function, hence the linear function has a faster convergence rate. However, the hyperbolic function is much smoother and has smaller oscillations in queue length. Due to the stochastic effects of arrival and departure rates in the queueing system, we define that when $\frac{q_i}{Q_{eq}} \in [0.9, 1.1]$, the system is in the stable state. Thus, assuming after n steps, the queue is stable, we rewrite (5) into

$$0.9Q_{eq} \leq \left(1 - \frac{\alpha k}{Q_{eq}}\right)^n q_0 + Q_{eq}\left[1 - \left(1 - \frac{\alpha k}{Q_{eq}}\right)^{n+1}\right] \leq 1.1Q_{eq} \tag{6}$$

Assuming $q_0 = M \times Q_{eq}$, using (6) we have

$$n \geq \frac{\log \frac{1}{10\left|M-1+\frac{\alpha k}{Q_{eq}}\right|}}{\log\left(1 - \frac{\alpha k}{Q_{eq}}\right)} \tag{7}$$

Now we show a simple example to calculate the convergence time for FERA. We assume N_0 is relatively large compared to the the actual number of flows in the network. Therefore, at the very beginning, $q_0 = 0$. Let us assume $Q_{eq} = 16$, $P_m = 1500$ B, $C = 10$ Gbps, $M = 0$, $T = 1$ ms and $k = 0.002$. Using equation (7), we have $n \approx 20$, then the convergence time $t = n \times T = 20$ ms. By increasing k, we can achieve shorter convergence time, for example, if $k = 0.008$, we only need $n \approx 4$ steps.

6.2 Convergence Time for BCN

Recall the AIMD algorithm implemented by BCN [5], at the i^{th} source, assume t_n is the time at n^{th} rate update event, then

$$r_i(t_{n+1}) = \begin{cases} r_i + G_i e(t_n) R_u & \text{if } e(t_n) > 0; \\ r_i[1 + G_d e(t_n)] & \text{if } e(t_n) < 0. \end{cases}$$

Here, $e(t_n)$ is the congestion index measured for time interval $[t_{n-1}, t_n]$. G_i is the additive increase gain, R_u is the increase rate unit and G_d is the multiplicative decrease gain.

By the above AIMD algorithm, we have the following discrete differential equation,

$$\begin{aligned} r_i(t_{n+1}) = & \underbrace{e_i(t_n)^\dagger [1 - |e_i(t_n)| G_d] r_i(t_n)}_{\text{Multiplicative Decrease}} \\ & + \underbrace{(1 - e_i(t_n)^\dagger)[r_i(t_n) + G_i |e_i(t_n)| R_u]}_{\text{Additive Increase}}, \end{aligned}$$

where

$$e_i(t_n)^\dagger = \begin{cases} 1, \text{if } e_i(t_n) < 0, \\ 0, \text{otherwise.} \end{cases}$$

Assume the absolute value $|e_i(t)|$ is independent of the sign of $e_i(t)$, the above equation can be rewritten as:

$$r_i(t_{n+1}) - r_i(t_n) = |e_i(t_n)| \{ G_i R_u - e_i(t_n)^\dagger (G_d r_i(t_n) + G_i R_u) \} \quad (8)$$

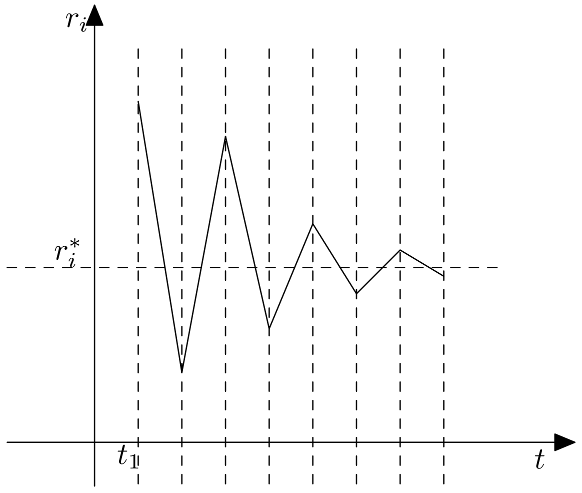


Figure 6.1: A sample path of rate convergence process in BCN.

Take the expectation on $e_i(t_n)^\dagger$ over time, generally we can assume $E[e_i(t_n)^\dagger] = \frac{1}{2}$ (In fact, this value is correlated with the parameters G_i, G_d and R_u). In other word, if we regard it as the expectation of the probability of generating negative BCN messages, in fairness state, $E[e_i(t_n)^\dagger] \approx \frac{1}{2}$, otherwise, the rate will either explode to infinity or zero. A simple justification from Fig.6.1 is that the rate is always oscillating around the fair share. Another observation is that the absolute difference between $r_i(t_{n+1})$ and $r_i(\infty)$ must vanish when time goes to infinity. Therefore we take expectations on (8) and rewrite it into the following one,

$$|r_i(t_{n+1}) - r_i(\infty)| = \{1 - E[|e_i(t_n)|]E[e_i(t_n)^\dagger]G_d\} \cdot |r_i(t_n) - r_i(\infty)|, \tag{9}$$

where we have

$$r_i^* := r_i(\infty) = \frac{G_i R_u(1 - E[e_i(t_n)^\dagger])}{E[e_i(t_n)^\dagger]G_d}. \tag{10}$$

Note that (8)-(10) are only an approximation on the convergence process of the rate, which is not rigorously accurate but provides us a way to lower bound the convergence time for BCN. A similar approach for loss-based TCP AIMD algorithm can be found in [12].

Based on the recursive relation in (9), denote $E|e_i(t_n)|E[e_i(t_n)^\dagger]G_d$ as $e_{i,n} \in (0,1)$ for convenience, we have

$$r_i(t) = r_i^* + (r_i(0) - r_i^*) \prod_{t_n \le t} (1 - e_{i,n}), \tag{11}$$

where $r_i(0) < r_i^*$.

Assume $e = \min\{e_{i,n}\}$, then

$$r_i(t_n) \ge r_i^* + \frac{r_i(0) - r_i^*}{n+1} \frac{1 - (1-e)^{n+1}}{e}. \tag{12}$$

Then assume averagely the source get one BCN message in T', we can have a lower bound on time to convergence $t = nT'$ by

$$r_i^* + \frac{r_i(0) - r_i^*}{n+1} \frac{1 - (1-e)^{n+1}}{e} \ge \beta r_i^*,$$

where $\beta = 0.9$ is defined to be the lower bound of the range of the fair state.

By $(1-x)^{n+1} \le 1-(n+1)x \le 1-(n+1)x^n, x < 1$, one has $\frac{1-(1-e)^{n+1}}{e} \ge (n+1)(1-e)^n$. Thus

$$n \ge \frac{\log(1-\beta)r_i^* - \log(r_i^* - r_i(0))}{\log(1-e)}. \tag{13}$$

For practical implementation of BCN, we use the well-tuned $G_d = 2.6 \times 10^{-4}$ [13]. Suppose there is a new flow injecting into the network, with initial rate 1.1 Gbps, then assume the other 3 flows has total throughput around 9 Gbps, which means that the network is slightly congested. Typically now, $e_i(t_n) \le 3$. Then using (13), it will take 4416 steps for the new flow to converge to fair share. Assume that BCN generates feedback one message per 100 packets for 10 Gbps link ($T' \approx 1.2$ ms), then the convergence time is around 0.529 seconds. Therefore the example shows an example about the poor convergence property of BCN. However, this case is not a problem for FERA because after one measurement interval, all the flows can send data with the same rate. The convergence to fairness will not be affected by network conditions.

Chapter 7

Enhancement to Switch Algorithm

Generally, the switch algorithm shown in Chapter 5 works well under good network conditions like continuous traffic pattern, short delays, etc. However, the real network traffic is very bursty and propagation delays are relatively long. In order to make FERA robust, we introduce several enhancements to help FERA to maintain stability and low queue length.

7.1 Exponential Moving Average

Since there are measurement noise and stochastic perturbations in the network, to make the rate control robust, we introduce a simple exponential moving average process on the advertised rate calculation. Assume that in the $(i-1)^{th}$ and i^{th} interval, the advertised rates are r_{i-1} and r_i, respectively. Then the new advertised rate is written as:

$$r_{i+1} = \lambda \frac{r_i}{\rho_i} + (1-\lambda) r_{i-1}, \tag{14}$$

where the constant $\lambda \in (0,1)$.

Proposition 2 With exponential moving average in (14), FERA can still achieve stability.

Without loss of generality, assuming $q_1 < q_2 < \dots < q_i < \dots$ and substituting (14) into (4), we have

$$q_{i+1} = q_i + \lambda\alpha(f(q_i) - 1) + (1 - \lambda)\frac{(A_{i-1} - C)T}{P_m}, \tag{15}$$

Note that $\frac{(A_{i-1}-C)T}{P_m} = q_{i-1} - q_{i-2}$, then

$$q_{i+1} = q_i + \lambda\alpha(f(q_i) - 1) + (1 - \lambda)(q_{i-1} - q_{i-2}). \tag{16}$$

With the linear queue control function (1), we have

$$q_{i+1} = \left(1 - \frac{k\lambda\alpha}{Q_{eq}}\right) q_i + (1 - \lambda)(q_{i-1} - q_{i-2}) + k\lambda\alpha, \tag{17}$$

Let $q'_i = q_i - Q_{eq}$, (17) can be rewritten as

$$q'_{i+1} = \left(1 - \frac{k\lambda\alpha}{Q_{eq}}\right) q'_i + (1 - \lambda)(q'_{i-1} - q'_{i-2}),$$

Then

$$\left(1 - \frac{k\lambda\alpha}{Q_{eq}}\right) q'_{i+1} < \left(1 - \frac{k\lambda\alpha}{Q_{eq}}\right) q'_i + (1 - \lambda)(q'_{i-1} - q'_{i-2}).$$

Since $0 < \frac{k\alpha}{Q_{eq}} < 1$, we have

$$\frac{q'_{i+1} - q'_i}{q'_{i-1} - q'_{i-2}} < \frac{1 - \lambda}{1 - \frac{k\lambda\alpha}{Q_{eq}}} < 1. \tag{18}$$

(18) shows that $\lim_{i\to\infty} q'_{i+1} - q'_i = 0$. In other words, $\{q_i\}$ is a Cauchy sequence and, therefore, it converges. Letting $i \to \infty$ at both sides of (14), we have $\lim_{i\to\infty} q_i = Q_{eq}$. Hence, with the moving average, the system is still stable.

Heuristically, this moving average process generally prevents the system from jumping too far away from the current position by using the historical information.

Furthermore, it helps overcome the effect of round trip delays. Assuming that the average control loop delay is $D < T$, then for i^{th} measurement interval, the average arrival rate is

$$A_i = \frac{N}{T}[r_i(T-D) + r_{i-1}D] = N\left(\frac{T-D}{T}r_i + \frac{D}{T}r_{i-1}\right). \qquad (19)$$

Generally with long control loop delay, the measurement of ρ_i will be *biased*. However, with the moving average, it can compensate the system in the reverse direction so that the advertised rate can be pulled back to the unbiased value. For example, if $r_{i-1} < r_i < r_{i+1}$, the arrival rate A_i is small compared with the value without delay, so is ρ_i. The switch still uses r_i to compute r_{i+1}, but actually in the i^{th} interval, the average rate of individual flow is less than r_i. Then the advertised rate computed without moving average is larger than the *expected* value which is free of delay. However, by the moving average, the advertised rate is smaller. In this sense, the moving average process is very helpful in reducing the effect of control loop delays.

7.2 Limited Increase

In order to avoid very high queue lengths, we put an upper bound on the rate increment. Assume initially the number of flows in the network are estimated by N_0, and if the advertised rate increases by Δr, then in the next measurement interval, the total input rate on the congested link increases by $N_0\Delta r$. When N_0 is large, it can cause the queue to build up very quickly. So it is helpful to limit $\Delta r \leq \delta$. Note that the smaller δ results in the longer convergence time. A further step to handling smaller δ is to exponentially enlarge it when the queue length is lower than some threshold. In particular, for each interval, we define the limited increase δ_i as follows. If at the measurement time, the queue length is less that Q_{eq}, set $\delta_i = \eta\delta_{i-1}$, and if the queue length is larger than Q_{sc}, set $\delta_i = \frac{1}{\eta}\delta_{i-1}$, where $\eta > 1$. In this way, if the actual number of flows in networks is n, we can get a good rate increment in approximately $\log_\eta(\frac{N_0}{n})$ steps.

7.3 Varying Capacity Enhancement

In some situations, link capacity can change suddenly. For example, if link aggregation is used to create a 10 Gbps link from 10 one-Gbps links and if nine of those one-Gbps links fail, the service rate can decrease suddently to 10%, i.e., 1 Gbps. FERA can cope with such variations in the link capacity. Suppose the link capacity at time t_{i-1} is C_{i-1} and at time t_i the link capacity changes to C_i. If $C_i < C_{i-1}$, we simply scale the advertised rate as

$$r_{i+1} = \frac{C_i}{C_{i-1}} \left[\lambda \frac{r_i}{\rho_i} + (1-\lambda) r_{i-1} \right], \tag{20}$$

where we assume the use of moving average. Note that we do not scale the advertised rate when $C_i \geq C_{i-1}$. There are two reasons for this. First, it is always better for the switch to be careful before increasing its rate since queue build up is very harmful. Second, if the link service rate increases severely, the queue can build up very easily simply because the scaling also magnifies the error between advertised rate and fair share. For example, assume the link service rate increases from 1 Gbps to 10 Gbps and there are 10 flows in the network. With link service rate of 1 Gbps, the system achieves fairness when each flow's rate is in $[95, 105]$ Mbps. Now for 10 Gbps link rate, if we scale these rates to $[950, 1050]$ Mbps, and all 10 flows change their rate to, say, 1050 Mbps, the queue can easily build up since the excess input of 50×10 Mbps in one measurement interval $T = 1$ ms is around 42 packets with a packet size of 1500 bytes. In other words, $[950, 1050]$ is actually not an admissible set of rate for system stability with 10 Gbps link.

7.4 Queue Control Enhancement with Heavy Traffic

Real world traffic is bursty, which can fill up the queue very quickly. One key observation is that when the initial queue length is larger than Q_{eq}, and at the equilibrium point (when every source sends out packets with fair share rate), the queue cannot approach Q_{eq} either. If the link load is equal or larger than 1, the queue can still build

γ	Throughput(Mbps)
0.98	9154.14
0.99	9329.22
1.00	9154.14

Table 7.1: FERA with heavy traffic.

up to infinity by the theory of heavy traffic. In order to cope with heavy traffic, when $q > Q_{eq}$, we use γC to replace the link capacity, where $0.95 < \gamma < 1$. Therefore, when the queue length is high and sources get the fair share, the queue can continue to drain. Simulation results with 100 CBR flows with 200 Mbps data rate going through one switch output queue with 10 Gbps service rate are shown in the following table. From Table 8.1, we see that $\gamma = 0.99$ is a good choice.

7.5 Equivalence of Multiplicative and Additive Queue Control

An alternative to multiplying the capacity by $f(q)$ is to subtract $f(q)$ from the capacity. Both of these will lead to reduction in capacity when the queue length is large. Simulation results[7] show that multiplicative control performs well. In the following, we show that the multiplicative and additive queue control are equivalent. Based on the fluid model, we adopt the form of additive queue control which is extensively studied in [6]. It is written as

$$r_{i+1} = r_i \left[1 + \frac{\alpha(\gamma C - A_i) - \beta \frac{q_i - Q_{eq}}{T}}{\gamma C} \right], \tag{21}$$

where α, β, γ are moving average parameters. $\gamma C - A_i$ is the available capacity to share among flows. If $\gamma C - A_i > 0$, there is spare capacity; Otherwise, the link is overloaded. $(q_i - Q_{eq})/T$ is used to drain the queue to Q_{eq} in one measurement interval. $\gamma < 1$ gives some additional space for the queue to drain as we discussed in last section to cope with heavy traffic. Note that large α helps in utilizing available spare bandwidth and large β helps in draining the queue aggressively.

Recall that the advertised rate update in FERA is

$$r_{i+1} = \frac{r_i}{\rho_i} = r_i \left(1 + \frac{1}{\rho_i} - 1\right). \tag{22}$$

Without loss of generality, assume $q > Q_{eq}$, then (3) can be rewritten as

$$r_{i+1} = r_i \left[1 + \frac{\frac{C}{A_i}(C - A_i) - \frac{C}{A_i}\frac{kT}{Q_{eq}}\frac{(q_i - Q_{eq})}{T}}{C}\right]. \tag{23}$$

Comparing (21) and (23), we can find that they are quite similar except for the moving average parameters. Note that $\frac{C}{A_i} < 1$ when there is congestion. Hence, there is no substantial difference between the multiplicative and additive queue control. In other words, since multiplicative queue control still follows the fluid model, the system will eventually converge to stable state given proper conditions.

7.6 Multistage Queue Control Function

When a switch is congested, the queue length in the switch is a big concern in data center applications. In this section, we further refine the queue control into a *multi-stage* function shown in Fig.7.1, which enables fast queue draining when the queue length is larger than a predefined threshold, such as $2Q_{eq}$. Here the n^{th} stage is defined as the piecewise control defined on $(nQ_{eq}, (n+1)Q_{eq}]$. For larger n, the queue drain is faster. For simplicity, only piecewise linear queue control functions are considered. It is straightforward to extend them into hyperbolic multistage queue controls. For 100 flows with average rate 200 Mbps injecting into one switch output queue with 10 Gbps service rate, the link utilization for the multistage queue control function is shown in Table 7.2. Both non-bursty Bernoulli UDP traffic[7], and bursty UDP traffic with Pareto ON/OFF bursts of 10 ms and shape parameter 1.5 are simulated. Here, we use $k = 0.002n$ for the n^{th} stage. Comparing with the previous results, we see that multistage queue control improves the link utilization. For the bursty traffic, the improvement is smaller than that for the continuous traffic.

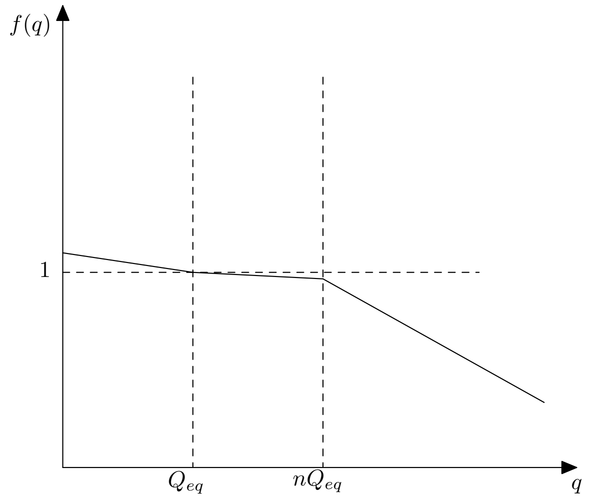


Figure 7.1: Multistage queue control function with $n = 2$

Queue Control	Throughput(Mbps)(Bernoulli)	Throughput(Mbps)(Pareto)
Single Stage	9329.22	9376.32
Multi Stage $n = 2$	9446.28	9366.78
Multi Stage $n = 3$	9814.68	9401.22

Table 7.2: FERA with single stage and multistage queue control, Bernoulli and Pareto traffic.

7.7 FERA and PAUSE mechanism

IEEE 802.3x PAUSE mechanism is a hop by hop flow control mechanism, which is used in DCEs to ensure zero loss when there is a sudden surge of link load due to link failure or traffic rerouting. When the queue length is larger than a threshold Q_{on}, the switch sends out PAUSE/ON to all its uplink neighbors. The neighbors stop transmitting packets. This in turn results in neighbor's buffers getting filled up and PAUSE/ONs are issued to their previous hops. Ultimately, PAUSE signal reaches the source end stations. When the queue length in the switch becomes lower than some predefined level Q_{off}, a PAUSE/OFF frame is sent out on the input ports to restart transmission of packets. Note that in each PAUSE interval, the number of dequeued packets is at most $L - Q_{off}$, where L is the buffer size in packets. Therefore, the

PAUSE interval t_P is

$$t_P = \frac{(L - Q_{off}) \times P_m}{C}.$$

For a typical DCE with 10 Gbps link, $Q_{on} = 90, Q_{off} = 80, P_m = 1500$ B and $L = 100$, each PAUSE interval lasts $\frac{20\times1500\times8}{10\times10^9} = 0.000024\ s$, which is much shorter than one measurement interval (typically 1 ms). Therefore, in one measurement interval, there could be multiple PAUSE intervals. Meanwhile, due to the effect of PAUSE mechanism, no packets are dropped. During that measurement interval, the average load is around 1, otherwise, the congestion has already been controlled. Note that the queue control function still take effects since the queue length at the measurement time is at least Q_{off}, which leads to $\rho > 1$. Typically for the queue control function shown in Fig. 5.2, $\rho \approx 1.4$. Clearly, after several measurement intervals, the advertised rate will be exponentially reduced to the equilibrium rate.

Chapter 8

Simulation Results

In this chapter, we provide some simulation results to support our claims in former sections. For extensive simulation results on FERA and comparisons, refer to [7].

8.1 Simulation Configuration

We used Network Simulator V2 (NS2)[14] for our simulations. Unless noted otherwise, all simulations presented in this paper use the following parameters and configurations. Link propagation delays are 0.5 μs, which are typical for optical fiber lengths of 100 m. Node processing delay is 1 μs. Link speeds are 10 Gbps. Drop-tail mechanism is used when buffers overflow. Generally we assume the switch output buffer size is 100 packets. The PAUSE ON/OFF thresholds are 90/80 packets, respectively. The FERA parameters are: $Q_{sc} = 80, Q_{eq} = 16, a = 1.1, b = 1.002, c = 0.1$, For the simple topology, $N_0 = 20$, for the large topology, $N_0 = 200$. The measurement interval $T = \tau = 1$ ms. All packets are 1500 bytes. The settings of BCN is standardized in [7].

8.2 TCP Flows and FERA

First, FERA can protect fragile sources with lower rates. We use the simple topology shown in Fig. 8.1, where sources SR_1 and SR_2 are reference sources with relative lower rates, whose sinks are DR_1 and DR_2, respectively. The reference sources SR_1

and SR_2 have one connection which periodically sends out 10 KB data, then idle for several microseconds and transmit again. CS is the core switch. $ES_1, \ldots, ES_6$ are edge switches, and $ST_1, \ldots, ST_4$ are TCP sources with bulk traffic, whose sink is DT. Each of $ST_1, \ldots, ST_4$ have 10 continuous connections simultaneously. In the simulation, TCP Reno with Selective Acknowledgements (SACK) is used. The maximum timeout for TCP is tuned to 1 ms to enable fast recovery from segment losses. Since the timeout period is very small, the interaction between transport layer and IP layer is negligible. For TCP Reno, the maximum receive window is set to 44 packets, which is approximately equivalent to a window of 64 kB.

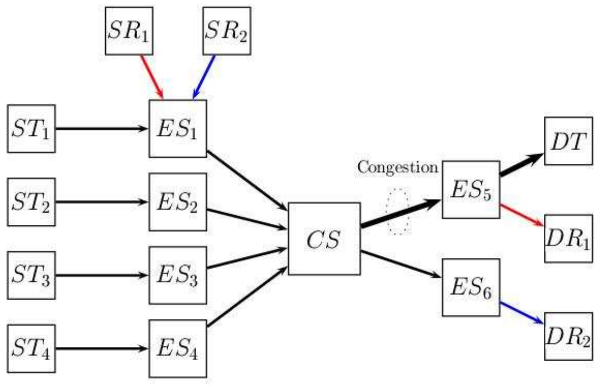


Figure 8.1: A simple topology.

The results with and without FERA are shown in Table 8.1 and 8.2 for a simulation run of 1 s.

From Table 8.1 and 8.2, it is seen that FERA improves the throughput of reference source 1 which suffers the congestion, and its flow completion time is shortened significantly. FERA presents a fairly allocated stable link to each source and so the variation in the TCP performance of the four sources is significantly reduced resulting in very high fairness. Also, FERA performs better than BCN, especially, FERA achieves perfect fairness considering the throughput variation between bulk TCP flows.

	Reference Flow 1		
CM	Throughput(Tps)	Throughput(Gbps)	FCT(μs)
None	556	0.060	1780.78
BCN	6686	0.578	133.51
FERA	6970	0.604	127.63
	Reference Flow 2		
CM	Throughput(Tps)	Throughput(Gbps)	FCT(μs)
None	16634	1.4398	59.11
BCN	16624	1.4390	59.16
FERA	16630	1.44	59.16

Table 8.1: Comparison of reference sources between None Congestion Management, BCN and FERA. (Tps means transaction per second, FCT means flow completion time)

CM	Average Throughput	Standard Deviation/Mean (%)	Link Utilization(%)
None	2.49	16.6%	99.9
BCN	2.35	16.8%	99.9
FERA	2.35	0.2%	99.9

Table 8.2: Comparison of bulk traffic between None Congestion Management, BCN and FERA.

8.3 Simple Symmetric Topology

The goal of this experiment is to show that FERA can achieve perfect fairness and stability compared with BCN. We start with a simple symmetric topology shown in Fig. 8.2. The traffic generation model is Bernoulli[13], which has traditionally been used in IEEE 802.1Qau group to model UDP traffic over Ethernet. If the average rate of Bernoulli UDP flow is r, packets are generated at rate $2r$ and then a Bernoulli trial with probability 0.5 is performed for each packet to decide whether it is dropped. Therefore, the instantaneous peak rate of Bernoulli flow with average rate r could be $2r$. In the following simulation, the average data rate for each of 4 flows is 5 Gbps. Four flows simultaneously are injected into the network at 5 ms. Then two flows goes away from 80 ms. In the simulation, the whole simulation duration is 100 ms. From the results, it is seen that the source throughput with FERA converges to the

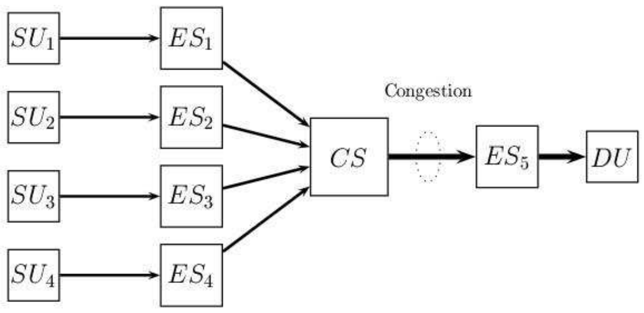
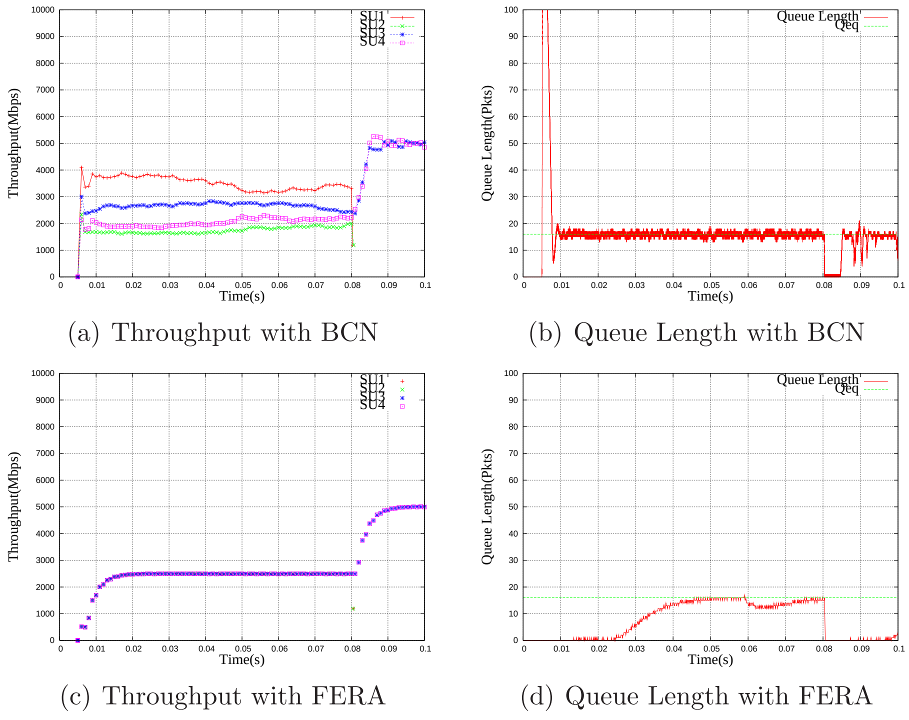


Figure 8.2: A simple symmetric topology.



(a) Throughput with BCN (b) Queue Length with BCN

(c) Throughput with FERA (d) Queue Length with FERA

Figure 8.3: UDP bernoulli traffic in simple symmetric topology.

fair share in around 10 ms, the queue goes to the Q_{eq} in much longer time, which somehow matches the analysis results in Section 6.1 when $k = 0.002$. However, as long as the source throughput is near the fair share, though the queue does not reach Q_{eq}, the link is fully utilized. In the same case, there is large difference between the throughput of each flow for BCN, which shows the unfairness of BCN in short time scale.

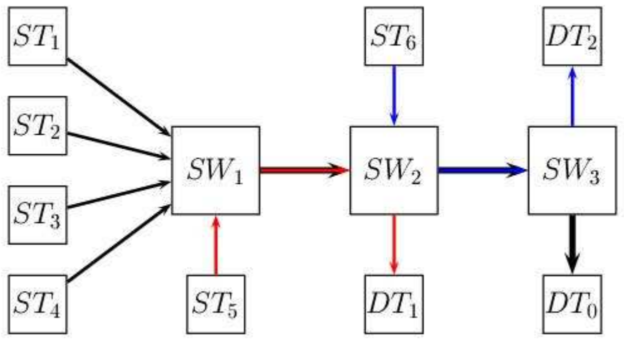


Figure 8.4: A parking lot topology

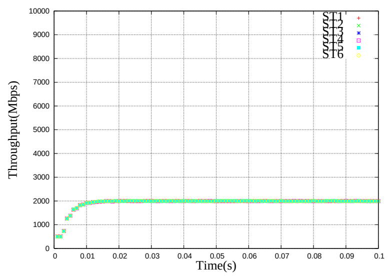
8.4 Parking Lot Topology

In this part, we show that for parking lot topology Fig. 8.4, FERA can achieve max-min fairness and stability. We still use the Bernoulli traffic. In Fig. 8.5, the results for both short and long control loop delay are presented, which clearly show FERA achieves max-min fairness and works well under long control loop delays. With long control loop delays, the oscillations in queue length are larger but still stable.

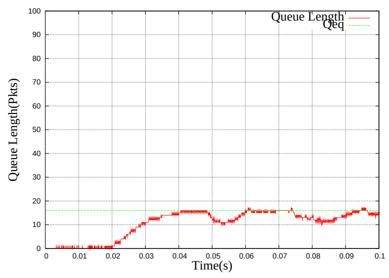
8.5 Large Symmetric Topology with Pareto Distributed Traffic

In this experiment, we simulate a configuration with a large number of flows and bursty traffic. We use the topology shown in Fig. 8.6, where $L = 25$. The traffic ON/OFF times are both Pareto distributed with an average of 10 ms. So all the 100 flows are very bursty, resulting in a severe test case for congestion management.

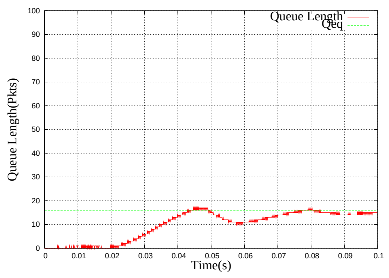
The following results show that FERA works very well with the bursty traffic. It achieves good fairness and stability of the source throughput. Even though there are oscillations in the queue length, the average queue length is still around Q_{eq}. This means that the system is robust with the bursty traffic. Also the maximum



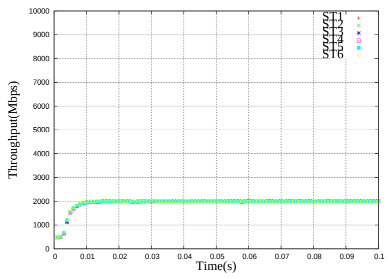
(a) Throughput



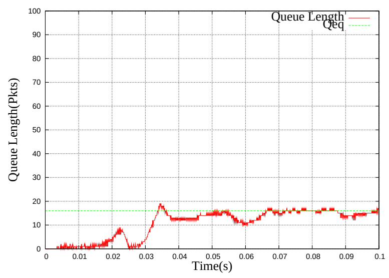
(b) Queue Length at SW_1



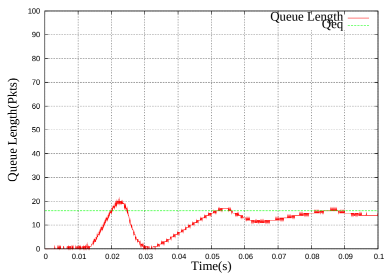
(c) Queue Length at SW_2



(d) Throughput



(e) Queue Length at SW_1



(f) Queue Length at SW_2

Figure 8.5: Simulation with parking lot topology: in(a)(b)(c), the one hop delay is 0.5 μs, in (d)(e)(f), the one hop delay is 50μs.

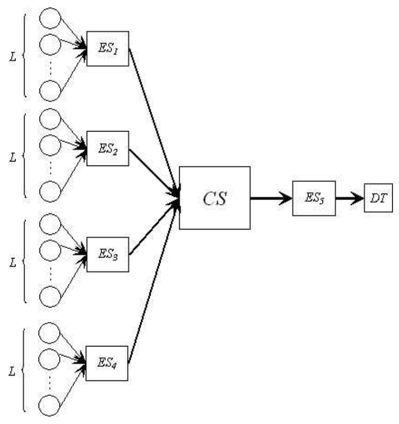
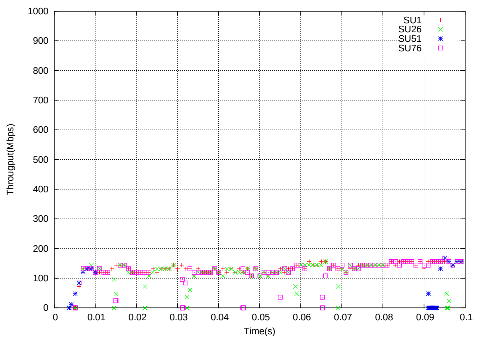
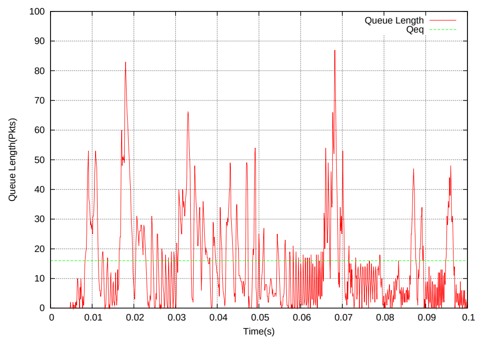


Figure 8.6: A simple symmetric topology.

queue length here is generally less than 90 packets, i.e., the PAUSE mechanism is not triggered.



(a) Throughput



(b) Queue Length

Figure 8.7: Pareto traffic in large symmetric topology.

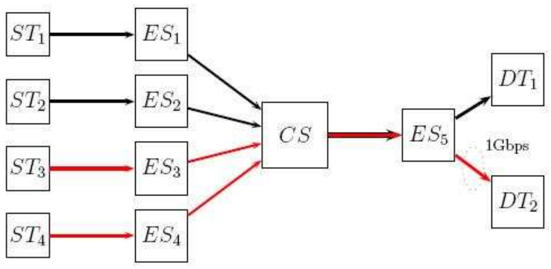
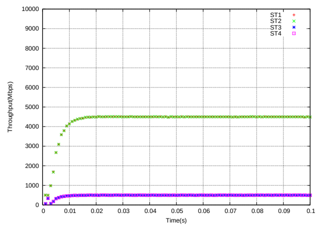


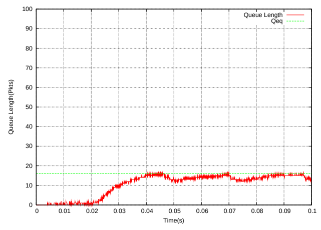
Figure 8.8: A simple asymmetric topology.

8.6 Asymmetric Topology and Multiple Congestion Points

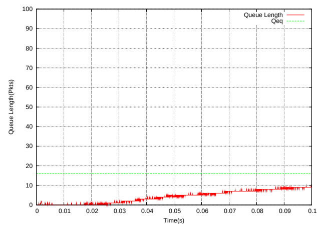
In the previous experiments, we assumed that all the links have the same capacity. This is, of course, not true in practice. Therefore, in this experiment, we simulate a scenario with an asymmetric topology in which some links in the network have low capacity, say, 1 Gbps. The topology we use is shown in Fig. 8.8. Now two congestion points are likely (the two right most links). The sink for ST_1 and ST_2 is DT_1 while the sink ST_3 and ST_4 is DT_2. From the throughput plot, it is seen that two flows have throughput of around 4.5 Gbps, and the other two have 0.5 Gbps. This is optimal. Note that the queue at ES_5 builds up much slowly, it is simply because



(a) Throughput



(b) Queue Length at CS



(c) Queue Length as ES_5

Figure 8.9: Throughput and queue length for asymmetric topology.

the link capacity is slow and in turn leads to longer convergence time. However, the throughput of the sources converges to fair share very fast.

8.7 Output Generated Hotspot Scenario

IEEE 802.1Qau group has selected many special configurations to compare different proposals. One such configuration is the so called ”one stage output generate hotspot” configuration shown in Fig. 8.10. The link capacity is 10 Gbps, the service rate of ST_0 decreases suddenly to 1 Gbps due to component link failures in the aggregated link. On each node, there are 9 flows destined to the other 9 nodes. For example, ST_1 has 9 flows sending data to $ST_0, ST_2, \ldots, ST_9$, respectively. The average rate of each flow is 0.945 Gbps. The throughput on the links, queue length and rate of individual flows are shown in Fig.8.11. It is clear that FERA can take care of the sudden link rate changes very quickly. Note that here one PAUSE event happens, but it takes effect only for 1 ms and so does not degrade the link throughput significantly.

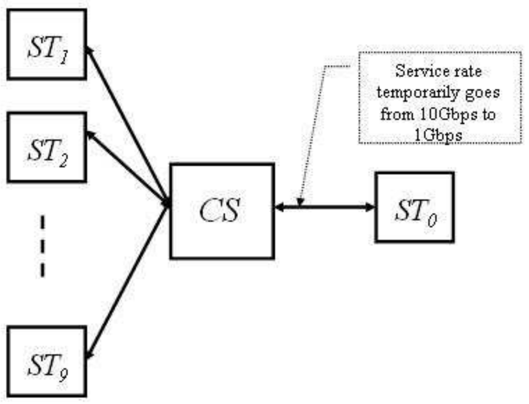
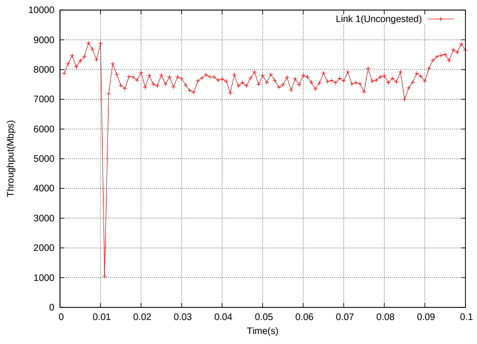
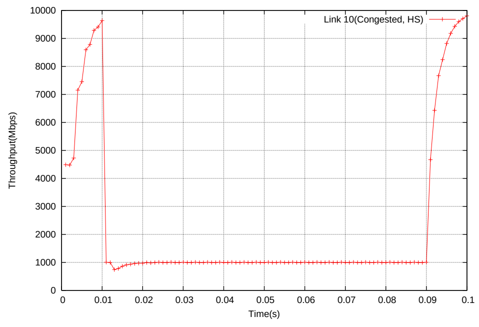


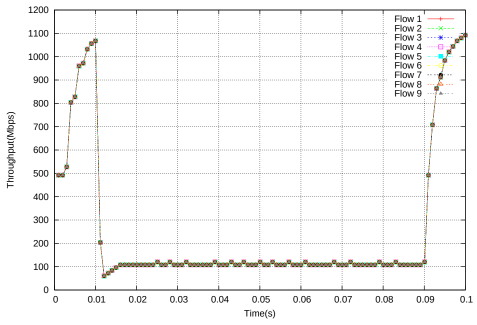
Figure 8.10: One stage output generated hotspot scenario.



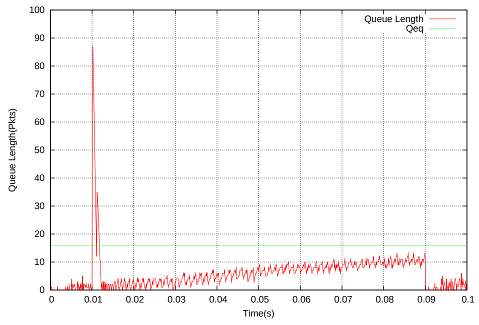
(a) Throughput on link from ST_1 to CS



(b) Throughput on link from CS to ST_0



(c) Throughput of each flow from CS to ST_0



(d) Queue length at port from CS to ST_0

Figure 8.11: One stage hotspot scenario, variational capacity.

Chapter 9

Conclusions

IEEE 802.1Qau group is developing a standard for congestion notification in data center Ethernet networks. The forward explicit congestion notification (FERA) scheme presented in this thesis is one of the proposed scheme discussed by the group. This thesis is the complete description of our proposal. In the thesis, we have presented both the analytical and simulation results for FERA mechanism and compare it with another scheme BCN. It is shown that FERA generally has better performance comparing BCN in convergence time and fairness.

References

[1] G. McAlpine, M. Wadekar, T. Gupta, A. Crouch, and D. Newell, "An architecture for congestion management in ethernet clusters," in *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 9*. Washington, DC, USA: IEEE Computer Society, 2005, p. 211.1.

[2] J. Santos, Y. Turner, and G. Janakiraman, "End to end congestion control for infiniband," *IEEE INFOCOM*, 2003.

[3] Y. Lu, R. Pan, B. Prabhakar, D. Bergamasco, V. Alaria, and A. Baldini, "Congestion control in networks with no congestion drops," *Allerton*, Sept 2006.

[4] D. M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Comp Networks and ISDN System*, 1989.

[5] J. Jiang and R. Jain, "Analysis of backward congestion notification (bcn) for ethernet datacenter applications," *IEEE INFOCOM Minisymposium*, May 2007.

[6] N. Dukkipati and N. McKeown, "Why flow-completion time is the right metric for congestion control," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 59–62, 2006.

[7] *802.1Qau - Congestion Notification*, IEEE 802.1au Working Group, http://www.ieee802.org/1/pages/802.1au.html.

[8] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control for fast long-distance networks," *IEEE INFOCOM*, 2004.

[9] R. Srikant, "The mathematics of internet congestion control," *Birkhauser*, 2004.

[10] B. Vandalore, R. Jain, R. Goyal, and S. Fahmy, "Design and analysis of queue control functions for explicit rate switch schemes," *IEEE IC3N*, Oct 1998.

[11] S. Kalyanaraman, R. Jain, S. Fahmy, R. Goyal, and B. Vandalore, "The erica switch algorithm for abr traffic management in atm networks," *IEEE/ACM Transactions on Networking*, Feb 2000.

[12] D. X. Xie, P. Cao, and S. H. Low, "Fairness convergence of loss-based tcp," *Paper Draft*, 2006.

[13] D. Bergamasco, "Cn-sim: A baseline simulation scenario," *IEEE 802.1 Meeting*, 2006.

[14] NS2, "Network simulator," *available at http://www.isi.edu/nsnam/ns/*, 1991.

[15] S. Golestani and S. Bhattacharyya, "End-to-end congestion control for the internet: A global optimization framework," *IEEE ICNP*, Oct 1998.

Vita

Jinjing Jiang

Date of Birth	March 7, 1981
Place of Birth	Huaiyin, China
Degrees	B.E. Electrical Engineering, July 2002 M.S. Electrical Engineering, January 2005
Related Publications	Jinjing Jiang, Raj Jain and Chakchai So-In, ”Congestion Management for Lossless Ethernet Operation,” submitted to Elsivier Computer Communications, 2008. Jinjing Jiang, Raj Jain and Chakchai So-In, ”An Explicit Rate Control Framework for Lossless Ethernet Operation,” to appear at IEEE ICC 2008. Jinjing Jiang and Raj Jain, ”Analysis of Backward Congestion Notification (BCN) for Ethernet In Datacenter Applications,” IEEE INFOCOM 2007 Minisymposium, May 2007.

May 2008

Note: Use month and year in which your degree will be conferred.

Short Title: Congestion Management for Lossless Ethernet OperationJiang, M.S. 2008