An Explicit Rate Control Framework for Lossless Ethernet Operation

Jinjing Jiang, Raj Jain and Chakchai So-In
Department of Computer Science and Engineering
Washington University in Saint Louis
{jinjing, jain, cs5}@cse.wustl.edu

***Abstract*—In this paper, the explicit rate control framework for Ethernet applications, especially data centers, is described. The framework guarantees zero packet drops at the congested switch and fast convergence to fair and stable state. In order to manage the congestion, design choices on 2-point and 3-point structure, the reactive and proactive signaling, explicit and implicit rate control are compared. Then the core component of the framework, queueing 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. Numerical results are provided to support the claims.**

I. Introduction

Ethernet networks are increasingly being used for data center applications involving storage and exchange of huge amounts of data. It is competing to replace Fiber Channel and Infiniband that are used to connect storage subsystems and that provide loss-free transmission. IEEE 802.1 standards group is developing mechanims to provide similar operation on Etherent networks.

Currently, Ethernet networks provide best effort service and depend on the transport layer protocols, like TCP, to respond to congestion. Even after 30 years of its invention, Ethernet networks run without congestion control in the data link layer though significant amount of work has been done on congestion control in TCP/IP networks. This may be acceptable for elastic applications but not tolerable for data center applications. TCP retransmissions and long timeouts could impair system performance significantly. Due to bursty nature of the traffic, switch buffers can overflow resulting in packet loss even when the average utilization is 50%. Without congestion control mechanisms, Ethernet networks are not very suitable for data center applications which are very sensitive to packet losses. Some of these applications use UDP and some don't even use IP (e.g., Veritas Cluster os[1]). Therefore, these applications cann't rely on TCP's congestion control mechansims.

In the literature, McAlpine and Wadekar [13] 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[6] describe a simple switch-based explicit congestion notirication (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 general Ethernet Congestion Management (ECM) scheme using Backward Congestion Notification (BCN) was proposed by Davide Bergasamo and his colleagues at Cisco and is described in [1][2]. The BCN messages contain current queue status. We proposed a forward explicit congestion notification mechanism (FECN) and argued in favor of explicit feedback of allowed rates to the sources. A modification to ECM, called Explicit Ethernet congestion management (E2CM) was proposed as combining some of the ideas of FECN and ECM. The fourth proposal is quantized congestion notificaiton (QCN)[7] in which ECN queue feedback is quantized to few bits and network provides only negative feedback. There are no positive feedback messages allowing sources to increase the rate. In stead, sources use a search algorithm to find the acceptable rate.

In this paper, we present new results on the performance of FECN and show that FECN provides better performance (fast convergence to stable and fair allocation). Also, using the Ethernet Pause mechanism along with FECN, good performance can be obtained even under severe congestion.

The remainder of the paper is organized as follows. Section II gives the system model and the components used for Ethernet link layer congestion management. Various design issues concerning the architecture are discussed in Section III. The queue control functions, which is the key part of the FECN mechanism, is described in Section IV. Section V analyzes the interaction between FECN and PAUSE mechanism. Then follows the conclusion.

II. System Model and Assumptions

For data center applications using Ethernet networks, end stations are connected to switches with high speed links, typically with line rate of 10 Gbps. For simplicity, we call these Data Center Ethernet (DCE) networks.

At the data link layer, in order to control the traffic, flows are rate regulated if there is congestion on the path. The rate regulation is realized by the hardware implementations of leaky bucket algorithm in network interface cards (NICs). Generally, the switches are equipped with measurement modules, which can detect the congestion by monitoring the queue length, the

load on the link, etc. The control messages and tags used for flow control are specially formatted packets sent by the switches in DCE. It is assumed that the congestion mechanism operates in a region consisting only of DCE-aware switches. The congestion control packets are not sent to or pass through legacy DCE-unware switches or hosts.

Fig. 1. A typical example of Ethernet for data center applications

An simple example of DCE is shown in Fig. 1. The network diameter of DCE is limited. Typically, the maximum path length is about 8 hops, with delay of 0.5 μs for each hop. The rate control for sources and switch congestion detection algorithms are expected to work in these limited environments.

III. Design Issues for the Architecture

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.

A. 2-Point and 3-Point Architectures

In our proposed FECN scheme, sources periodically generate probe packets that are modified by the switches along the path and then 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 a 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 2.

Fig. 2. The Architecture for Congestion Notification

Both ECM and QCN use 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 FECN, the sources do not have to keep track of congestion points and can increase their rate to the received feedback immediately. [The bit-based QCN was 3 point architecture but the received feedback was not explicit].

B. 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. 2, 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. This is reported in [3].

C. Explicit and Implicit Rate Control

ECM, 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, FECN 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 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 [4], it is reported that FECN's overhead is only around one tenth of the overhead of other schemes. Moreover, in [11], it is shown that FECN can converge to fair and stable state much faster than ECM does, simply because AIMD-like algorithms can achieve fairness only in the long term sense, which means that the transient time of the system is long [14].

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

IV. FECN Queue Control

In the previous sections, we have shown that FECN has many merits over other mechanisms. In this following section we answer the question - how FECN bounds the maximum delay and jitter in the Ethernet. Queue control plays a key role in the task of congestion management in FECN [11].

This paper discusses the queue control design and other queue related enhancements in depth.

First, we summarize the basic switch operations as follows. The switches perform time-based measurement to monitor the load and queue length. Let us denote the sequential measurement time instants as $t_0, t_1, t_2, \ldots, t_i$, where $T = t_i - t_{i-1}$ is the fixed measurement interval. See Fig.3. Then, the following

Fig. 3. Illustration of rate calculation

variables are 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 and P_m is the packet size. Note that we measure the queue length q_i by number of packets and use fixed size packets. However, FECN can handle random packet sizes simply by measuring queue lengths in bytes. In brief, the notation rules are as follows: 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.

- 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 probes, then $r = r_{i+1}$.

Note that in the last measurement interval, A_i is the sum of rates 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 to ensure that the queue length is kept at a constant level. Since in each measurement interval, the advertised rate is updated in a multiplicative fashion, we call our queue control a *multiplicative* control.

The analytic formula for the hyperbolic queue control function, which is the default in FECN and ERICA[15], is written as

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

where q is the instantaneous queue length in the switch measured by packets. a, b, c are constants.

Fig.4 plots the above function with parameters $a = 1.002, b = 1.1, c = 0.1$ and $Q_{eq} = 16$. We have the following observations:

- If $q < Q_{eq}$, $f(q)$ is only slightly larger than 1;
- If $q \ge Q_{eq}$, the hyperbolic function is very close to the linear control $f(q) = -0.00375q + 1.06 = -0.00375(q - Q_{eq}) + 1$;
- For 100 packet buffer size, $c = 0.1$ does not take any effect. Generally, it helps for large buffers and large Q_{eq}.

Fig. 4. queue control function, $a = 1.002, b = 1.1, c = 0.1$

For $q < Q_{eq}$, $f(q) > 1$ and $\rho = \frac{A}{f(q)C}$. Therefore $f(q)$ reduces the estimated load level and leads to an aggressive increment of advertised rate. If $f(q) = 1$, it is possible that the queue never approaches Q_{eq}, since the input rate cannot be larger than the link capacity and the initial queue length is always far less than Q_{eq}. Furthermore, for $q > Q_{eq}$, generally $f(q) < 1$. Then the queue control function attempts to aggressively decrease the advertised rate. In the next section, we prove the stability of this queue control.

A. Stability Criteria for Queue Control

The hyperbolic queue control function, though it has a smooth control on queue length, is nonlinear. So it is not trivial to analyze its fluid model. However, we can use the linear queue control function to get an approximation (or lower bound) to study the conditions for system stability.

Suppose $f(q)$ is a linear function written in the following form

$$f(q) = 1 - k\frac{q - Q_{eq}}{Q_{eq}}, \quad (2)$$

where k is some constant. Note that for $q \le Q_{eq}$ and $q > Q_{eq}$, different k should be applied. Without loss of generality, we only discuss the choice of k for $q > Q_{eq}$. Similar results for $q \le Q_{eq}$ can be obtained easily.

The fluid model gives

$$q_{i+1} = q_i + \frac{(Nr_{i+1} - C)T}{P_m} = q_i + \alpha(f(q_i) - 1), \quad (3)$$

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

Thus, a closed form solution to q_i is written as

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

Then

$$q_i = \left(1 - \frac{\alpha k}{Q_{eq}}\right)^i q_0 + \alpha k \sum_{j=0}^{i} \left(1 - \frac{\alpha k}{Q_{eq}}\right)^j. \quad (4)$$

With a sufficient condition that $0 < \frac{\alpha k}{Q_{eq}} < 2$, $\lim_{i\to\infty} q_i = Q_{eq}$. Note that this sufficient condition is the key criterion in choosing the queue control function parameters given the length of measurement interval. For example, suppose $C = 10$ Gbps and $T = 1\ ms$, the sufficient condition gives $k \le \frac{Q_{eq}}{\alpha} = 0.0384$. Note that in FECN, a moving average process is done on advertised rate, which further relaxes the conditions for k to achieve system stability.

γ	Throughput(Mbps)
0.98	9154.14
0.99	9329.22
1.00	9154.14

TABLE I
FECN WITH γ

B. 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 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 I, we see that $\gamma = 0.99$ is a good choice.

C. 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[5] 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 [12]. It is written as

$$r_{i+1} = r_i \left[1 + \frac{\alpha(\gamma C - A_i) - \beta \frac{q_i - Q_{eq}}{T}}{\gamma C} \right], \qquad (5)$$

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 FECN is

$$r_{i+1} = \frac{r_i}{\rho_i} = r_i \left(1 + \frac{1}{\rho_i} - 1 \right). \qquad (6)$$

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

$$r_{i+1} = r_i \left[1 + \frac{\frac{C}{A_i}(C - A_i) - \frac{C}{A_i}\frac{kT}{Q_{eq}}\frac{(q_i - Q_{eq})}{T}}{C} \right]. \qquad (7)$$

Comparing (5) and (7), 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.

D. 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 *multistage* function shown in Fig.5, 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

Fig. 5. Multistage queue control function with $n = 2$

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 II. Both non-bursty Bernoulli UDP traffic[3], 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.

V. FECN AND PAUSE MECHANISM

IEEE 802.3x PAUSE mechanism is a hop by hop flow control mechanism, which is used in DCE 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

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

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 II
FECN WITH SINGLE STAGE AND MULTISTAGE QUEUE CONTROL, BERNOULLI AND PARETO TRAFFIC

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. 4, $\rho \approx 1.4$. Clearly, after several measurement intervals, the advertised rate will be exponentially reduced to the equilibrium rate.

VI. CONCLUSION

IEEE 802.1Qau group is developing a standard for congestion notification in Data Center Ethernet. Various design aspects are discussed in this paper to show that the forward explicit congestion notification (FECN) scheme presented in [11], is the better choice considering various system performance metrics and the complexity of implementation. This paper explains the design philosophy of our proposal. In addition, we have presented a thorough treatment on the queue control, which is the key for low-loss congestion management. Analytical and numerical results for different queue control functions and several enhancements are discussed. Finally, the analysis on the interaction between PAUSE mechanism and FECN scheme is presented, which together guarantee the convergence of the system and zero frame loss.

REFERENCES

[1] D. Bergamasco, "Data Center Ethernet Congestion Management: Backward Congestion Notification," *IEEE 802.1 Meeting*, May 2005.

[2] D. Bergamasco and R. Pan, "Backward Congestion Notification Version 2.0," *IEEE 802.1 Meeting*, September 2005.

[3] D. Bergamasco, "CN-SIM: A Baseline Simulation Scenario", *IEEE 802.1 Meeting,* 2006.

[4] D. Bergamasco, "CN-SIM: A Single Stage Output Generated Scenario", *IEEE 802.1 Meeting,* 2007.

[5] B. kwan and J. Ding, "Preliminary Simulation Results on FECN In Symmetric Topology w/Single Hot Spot Scenario", *IEEE 802.1 Meeting,* March 2007.

[6] J.R. Santos, Yoshio Turner, G. Janakiraman, "End to end congestion control for Infiniband," *IEEE INFOCOM*, 2003.

[7] P. Prabhakar, "QCN,"*IEEE 802.1 Meeting,* 2007.

[8] IBM Zurich Lab, "E2CM,"*IEEE 802.1 Meeting,* 2007.

[9] J. Jiang and R. Jain, "Analysis of Backward Congestion Notification (BCN) for Ethernet Datacenter Applications," *IEEE INFOCOM Minisymposium*, Anchorage, Alaska, May 7-11, 2007.

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

[11] J. Jiang, R. Jain, and C. So-In, "Congestion Management for Ethernet In Datacenter Applications Using Forward Explicit Rate Notification," submitted, 2007.

[12] Nandita Dukkipati and Nick McKeown, "Why Flow-Completion Time is the Right Metric for Congestion Control," *ACM SIGCOMM Computer Communication Reiview*, Jan 2006.

[13] G. Mcalpine, M. Wadeker et. al., "An architechture for congestion management in Ethernet clusters," IEEE International Parallel and Distributed Processing Symposium, 2005.

[14] D.X. Xie, P. Cao, and S.H. Low,"Fairness Convergence of Loss-based TCP," Paper Draft, Caltech 2006.

[15] Bobby Vandalore, Raj Jain, Rohit Goyal and Sonia Fahmy, "Design and Analysis of Queue Control Functions for Explicit Rate Swich Schemes," *IEEE IC3N*, Oct. 1998.

[16] IEEE 802.1au Work Group, http://www.ieee802.org/1/pages/802.1au.html.