Sliding Mode Congestion Control for Data Center Ethernet Networks

Wanchun Jiang, Fengyuan Ren, Ran Shu, Chuang Lin
Tsinghua National Laboratory for Information Science and Technology, Beijing 100084, China
Dept. of Computer Science Tsinghua University, Beijing, China 100084
Email: {jiangwc, renfy, shuran, chlin}@csnet1.cs.tsinghua.edu.cn

***Abstract*—Recently, Ethernet is being enhanced as the unified switch fabric of data centers, called Data Center Ethernet. The end-to-end congestion management is one of the indispensable enhancements, and Quantized Congestion Notification (QCN) has been ratified to be the standard. Our experiments show that QCN suffers from the oscillation of the queue at the bottleneck link. With the changes of system parameters and network configurations, the oscillation may become so serious that the queue is emptied frequently. As a result, the utilization of the bottleneck link degrades. Theoretical analysis shows that QCN approaches to the equilibrium point mainly through the sliding mode motion. But whether QCN enters into the sliding mode motion also depends on both system parameters and network configurations. Hence, we present the Sliding Mode Congestion Control (SMCC) scheme, which can drive the system into the sliding mode motion under any conditions. SMCC benefits from the advantage that the sliding mode motion is insensitive to system parameters and external disturbances. Moreover, SMCC is simple, stable and has short response time. QCN can be replaced by SMCC easily since both of them follow the framework developed by the IEEE 802.1 Qau work group. Experiments on the NetFPGA platform show that SMCC is superior to QCN, especially in the condition that the traffic pattern and the network state are variable.**

***Index Terms*—Data Center Ethernet, Sliding Mode Motion, Quantized Congestion Notification**

I. Introduction

Currently, there is substantial interest in enhancing Ethernet as the unified switch fabric for TCP/IP, Storage Area Networks (SANs) and High Performance Computing (HPC) networks in data centers [1], [2]. The IEEE 802.1 Data Center Bridging task group [1] is standardizing these enhancements to fill the performance gap between the traditional Ethernet and the unified switch fabric. The enhanced Ethernet is called Data Center Bridging (DCB), Data Center Ethernet (DCE) or Converged Enhanced Ethernet (CEE).

Congestion occurs when links are oversubscribed and traffic is excessive. In data center networks (DCNs), the link oversubscription ratio is large[3], and the traffic is highly bursty [4]. Hence, congestion naturally happens frequently in DCNs. Although the dominating transport layer protocol TCP involves a long-tested congestion control mechanism, it is optimized for the long-range Internet rather than the short-range DCNs. There are also other traffics running over the Ethernet without congestion management (CM) strategies, notably the streaming media using UDP. Thus, the native Ethernet CM scheme is needed in DCNs. On the other hand, techniques such as FCoE [5] and RoCEE [6] make efforts to accommodate traffics of both SANs and HPC to Ethernet at present. It would be more economical to deploy a uniform CM scheme on the link layer for all the traffics in DCNs.

In Mar. 2010, the Quantized Congestion Notification (QCN), developed by the IEEE 802.1 Qau work group, has been ratified to be the standard for the end-to-end CM of DCE [1]. Nowadays, QCN has been implemented in devices, such as Cisco Nexus 7000 Series [7] and FocalPoint FM6000 [8]. Historically, QCN is heuristically designed, drawing on the experiences of CM in the Internet. However, the environment of DCE differs from that of the Internet. Moreover, QCN involves segmented nonlinearity, which is intractable through classic linear analytical method. And its performance evaluations are conducted mainly by simulations and experiments. As a result, there are only a few theoretical results on QCN. In a word, QCN has not been understood thoroughly. In this paper, we investigate the end-to-end CM scheme for DCE. More specifically, we make the following contributions.

We reveal some drawbacks of QCN through theoretical analysis and experiments. The first drawback is that QCN suffers from the oscillation of the queue at the bottleneck link, and its performance depends on both parameters setting and network configurations. When system parameters or network configurations change, the oscillation may become so serious that the utilization of the bottleneck link degrades, especially in the condition that the switch buffer is shallow in DCE. Secondly, QCN approaches to the equilibrium point mainly through the sliding mode motion. However, whether the sliding mode motion occurs in the QCN system also depends on both system parameters and network configuration. QCN may fail to reach the equilibrium point via the sliding mode motion, and become unstable in certain conditions.

We design the Sliding Mode Congestion Control (SMCC) scheme for DCE. SMCC follows the framework of CM defined by the IEEE 802.1Qau work group, namely, QCN can be replaced by SMCC easily. Furthermore, SMCC is superior to QCN in the following aspects.

- QCN happens to utilize the sliding mode motion. But it is mainly heuristically designed and can't ensure the system enter into the sliding mode motion in any conditions. The most significant advantage of the sliding mode motion is that the system behaviors are insensitive to system parameters and external disturbances. To utilize this advantage, SMCC is designed such that the system can enter into the sliding

978-1-4673-0775-8/12/$31.00 ©2012 IEEE

mode motion starting from any states under any network configurations. Hence, SMCC is stable and robust.

- SMCC is much simpler than QCN. Thus, the hardware implementation of SMCC is more economical.
- SMCC is more responsive than QCN. The core rate adjustment algorithm of QCN is originated from BIC-TCP [10]. It feeds the weighted sum of the queue length and its derivative back, and does binary search for rate increase without using feedback information. However, BIC-TCP is designed for the environment of large Bandwidth Delay Product (BDP). Though the bandwidth of current Ethernet is also large, the BDP is small due to the low propagation delay in DCE. In SMCC, the queue length and its derivative are involved in the feedback packet separately. Hence, more feedback information about the congestion is obtained, though the overhead is increased slightly. With feedback information for both rate decrease and rate increase, SMCC is more responsive.

We evaluate SMCC on the NetFPGA platform [11] since both QCN and SMCC are designed for hardware implementation. Experimental results indicate that SMCC is stable, has short response time and can adapt to the changes of system parameters and network configurations in DCE.

The rest of this paper is organized as follows: Section II introduces the background. And then the motivation behind the work is presented in Section III. Subsequently, Section IV describes the design of the SMCC scheme and Section V evaluates SMCC on the NetFPGA platform. Finally, Section VI concludes this paper.

II. Background

A. Congestion Management in Data Center Networks

There exists a rich history of design and control-theoretic analysis of congestion management for the Internet [12], [13], [14]. However, there are some significant differences in DCE.

- No per-packet ACK in the Ethernet. Thus, it is hard to estimate the round trip time, and the congestion control algorithm can't be automatically self-clocked like TCP.
- The traffic is highly bursty, namely they can potentially arrive at the speed of the Network Interface Cards (NICs).
- The switch buffer size is much smaller than the router buffer size.
- The link lengths are small in data center (within $500m$), which implies that the propagation delay is small (within $3\mu s$). Therefore, the BDP is small in DCE.

Furthermore, the CM scheme faces additional requirements for DCE being the unified switch fabric in DCNs.

Simple: Algorithm should be simple enough to be implemented completely in hardware to handle traffics at the speed of $1Gbps$ or $10Gbps$ in DCE.

Losslessness: Link may be paused and packet can not be dropped in data center. Because upper layer protocols for storage traffic and HPC traffic rely on no frame loss to achieve low processing overhead and high performance.

Low Delay: To carry the HPC traffic over Ethernet, DCE should keep controllable low delay.

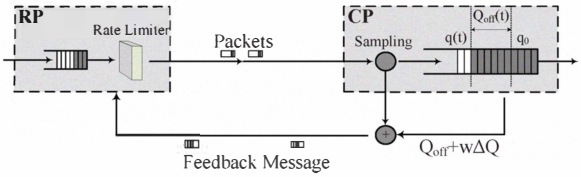


Fig. 1. Current Framework of the CM Mechanism in 802.1Qau

These special environment and requirements specialize the design of the CM scheme in DCE. The IEEE 802.1 Qau work group [15] have worked on the end-to-end CM scheme of DCE for several years. Four proposals have been released up to now, including Backward Congestion Notification (BCN) [16], Forward Explicit Rate Advertising (FERA) [17], Explicit Ethernet Congestion Management (E2CM)[18] and QCN[19]. In these proposals, various methods used in traditional CM mechanisms, such as explicit feedback v.s. implicit feedback and rate based load sensor v.s. queue based load sensor, are compared. At last, proper methods are embedded into the final framework and QCN is adopted as the final standard. However, the rate adjustment algorithm of QCN is heuristically designed and sensitive to both system parameters and network configurations. For certain network configuration, although the proper parameters setting of QCN can be obtained through experiments and simulations, the limit experiments and simulations hardly cover the unlimited network configurations. Hence, we intend to develop a new end-to-end CM scheme for DCE, which is insensitive to the changes of system parameters and network configurations, in this paper.

B. Framework of the Current CM Scheme

The current framework of the CM scheme developed by the IEEE 802.1 Qau work group is composed of two parts as shown in $Fig.1$.

The switch, or the Congestion Point(CP) The task of CP is to detect congestion, generate feedback packets and send them to sources.

The source, or the Reaction Point(RP) The goal of RP is to adjust the sending rate according to the feedback information.

Generally, CP monitors the queue length and "samples" incoming packets periodically with probability p to generate the feedback packets. The feedback information F_b representing the congestion state consists of two parts: the current offset of the queue length ($Q_{off} = q(t) - q_0$) and the variance of the queue length in a sampling interval ($\Delta Q = q(t) - q_{old}$), where q_0 is the target queue length and q_{old} is the queue length at the latest sampling. F_b is given by

$$F_b = -(Q_{off} + w * \Delta Q) \tag{1}$$

where w is a weight. RP receives the feedback packet and then adjusts the sending rate according to the feedback information involved in the packet. The rate adjustment is achieved by implementing the rate limiter at the edge switch or the NICs.

Historically, BCN uses the feedback information for both rate decrease and rate increase. There is also a CPID flag involved in the feedback packets, which univocally identify the congestion point. BCN responds to the congestion using a modified Additive Increase and Multiplicative Decrease (AIMD) algorithm, which is defined as follows:

$$r \leftarrow \begin{cases} r(1+G_dF_b) & when \quad F_b < 0 \\ r + G_iR_uF_b & when \quad F_b > 0 \end{cases} \tag{2}$$

where G_d is a constant chosen such that $G_d|F_{bmax}| = \frac{1}{2}$, i.e., the sending rate is decreased no more than 50% each time, G_i is the factor of rate increase and R_u is the unit of rate increase. The rate decrease algorithm of QCN is the same as BCN. But QCN employs a self-increasing algorithm for rate increase, similar to BIC-TCP. Thus, QCN only needs to generate the feedback packets involving negative feedback information and doesn't need to identify the congestion point. Let R denote the sending rate just before the arrival of the latest feedback packet. The rate increase algorithm of QCN is as follows.

Fast Recovery (FR) Immediately after the Rate Decrease (**RD**), RP enters into the state of **FR**. FR persists 5 cycles and the time length T of each cycle is set to be the time of sending $150KB$ data. At the end of each cycle, R keeps unchanged and r is updated by

$$r \leftarrow \frac{1}{2}(r+R). \tag{3}$$

Active Increase (AI) After the process of FR, RP enters into the **AI** state to probe for more available bandwidth. In the state of AI, R and r are updated at the end of transmitting 75KB data, namely the the time length of each cycle in the AI state is half of that in the FR state.

$$\begin{cases} R \leftarrow R + R_{AI} \\ r \leftarrow \frac{1}{2}(r+R) \end{cases} \tag{4}$$

where R_{AI} is the constant unit of rate increase.

We only present the core mechanisms of both BCN and QCN above. More details can be found in [15]. In general, QCN can be considered as an improved version of BCN. More specifically, QCN is more aggressive than BCN during the period of rate increase, and obtains an extra benefits of reducing the number of feedback packets. The FR mechanism is borrowed from the BIC-TCP algorithm, which is an improvement to the AIMD algorithm for the network with high BDP. The AI mechanism of QCN is a heuristical improvement of BIC-TCP. QCN still has other add-on improvements: R_{AI} is replaced by a larger constant R_{HAI} when the sending rate at the state of AI is really large, the Extra Fast Recovery mechanism and the Target Rate Reduction mechanism [20]. Obviously, QCN is more complex than BCN.

III. Motivation

To achieve the ideal performance, QCN adopts several additional mechanisms, most of which are heuristically designed, and lacks of theoretical analysis. To fill the gap, this section reports our empirical study of QCN by experimental observations and theoretical analysis. More precisely, we observe some interesting phenomena in experiments and obtain some insights through theoretical analysis. They serve as the motivation of designing the SMCC scheme for DCE.

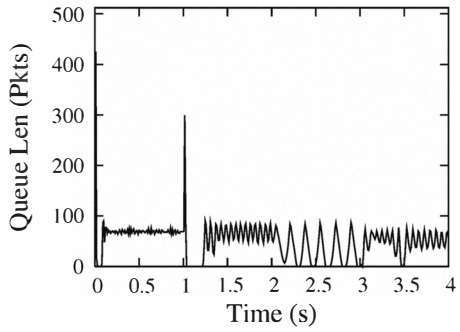


Fig. 2. Queue length of QCN at the bottleneck link

A. Experimental Study on QCN

To study the QCN mechanism, we implement it on the NetFPGA [11] platform. To explore the performance of QCN with the changes of system parameters and network configurations, we conduct experiments in the dumbbell topology. The system parameters and configurations of QCN are the same as that in [20]. The link capacity is $500Mbps$. The switch buffer size is $512KB$. Parameters for CPs are $w = 2$, $p = 0.01$ and $q_0 = 64KB$. F_b is quantized to $6bits$ in the feedback packets. Parameters for RPs are $G_d = \frac{1}{128}$, $R_{AI} = 1Mbps$. At the beginning of the experiment, two RPs start long-lived flows at the speed of $1Gbps$, and T is the time of sending $15KB$ data. Then, an extra flow of size $750Mbps$ preempts the bandwidth of the bottleneck link at the 1st second. Subsequently, parameter T is reset to be the time of sending $150KB$ data at the 2nd second. Finally, the extra flow stops at the 3rd second.

The evolution of queue length at the bottleneck link are shown in $Fig.2$. In the 1st seconds, the queue reaches the stable state quickly, chattering slightly. However, the queue oscillates after the bandwidth shared by RPs is preempted at the 1st second. After the parameter T is changed at the 2nd second, the queue oscillates so seriously that it becomes empty frequently. Accordingly, the link utilization heavily degrades. Note that, the network configuration in the 2nd second is the same as that in the 3rd second except for parameter T. Hence, the change of parameter T results in the unstable queue. After the extra flow stops at the 3rd second, the queue only seldom becomes empty. Comparing the evolution of the queue length either between the 1st second and the 2nd second or between the 3rd second and the 4th second, we know that the change of bandwidth also affects the performance of QCN. Similar results can be obtained when some other parameters or network configurations are changed. The corresponding experiments are omitted due to the limited space.

In total, QCN may become inefficient with heavy oscillation of the queue at the bottleneck link, when system parameters

and network configurations change.

B. Theoretical Analysis of QCN

Concerning on the bottleneck link, we build a fluid-flow model for QCN. Let $r(t)$ denote the sending rate of each source, since sources are always homogeneous due to symmetrical network topologies such as Fat-Tree [3] and the special computing paradigm such as Map-Reduce in data centers. We neglect the delay since the propagation delay is relatively small and then have

$$\frac{dq(t)}{dt} = Nr(t) - C \tag{5}$$

and

$$\Delta Q = \Delta t \frac{dq(t)}{dt} = \frac{1}{pC}[Nr(t) - C] \tag{6}$$

at the bottleneck link. Thus, equation (1) can be rewritten as

$$F_b(t) = -[q(t) - q_0] - \frac{w}{pC}[Nr(t) - C] \tag{7}$$

where N is the number of active flows sharing the bottleneck link, C denotes the capacity of the bottleneck link and other variables are defined in Section II. The rate adjustment algorithm of QCN can be modeled by

$$\begin{cases} \frac{dr(t)}{dt} = G_d F_b(t) r(t) & RD \\ \frac{dr(t)}{dt} = -\frac{r(t)}{2T} + \frac{R(0)}{2T} & FR \\ \frac{dr(t)}{dt} = -\frac{r(t)}{T} + \frac{R(0)}{T} + \frac{2R_{AI}t}{T^2} & AI \end{cases} \tag{8}$$

where $R(0)$ is the target sending rate for fast recovery. For the sake of simplicity, we make a substitution

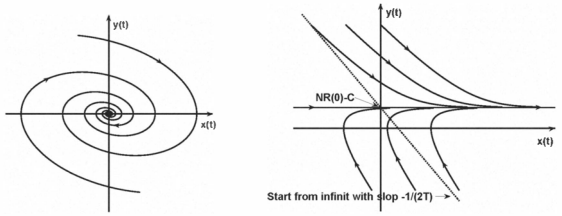
$$\begin{cases} x(t) = q(t) - q_0 \\ y(t) = Nr(t) - C \end{cases} \tag{9}$$

In this way, $x(t)$ denotes the offset of the queue length to the target point, $y(t)$ is associated with the sending rate and the equilibrium point of the system is transferred to $(x(t), y(t)) = (0, 0)$. Combining (5), (7) and (8), the differential equations describing the rate decrease subsystem of QCN can be obtained.

$$\begin{cases} \frac{dx(t)}{dt} = y(t) \\ \frac{dy(t)}{dt} = -G_d[x(t) + \frac{w}{pC}y(t)][y(t) + C] \end{cases} \tag{10}$$

Equation (10) means that the rate decrease subsystem of QCN is nonlinear, namely this multiple decrease method is different to that in the classic AIMD algorithm. Lyapunov has shown that the stability and the behaviors of the nonlinear system, such as (10), in the neighborhood of a singular point can be found from the linearized version of nonlinear differential equations about the singular point [21]. The linearized version of (10) about the singular point, i.e. the equilibrium point $(x(t), y(t)) = (0, 0)$, is

$$\begin{cases} \frac{dx(t)}{dt} = y(t) \\ \frac{dy(t)}{dt} = -G_d C x(t) - \frac{G_d w}{p} y(t) \end{cases} \tag{11}$$



(a) The trajectories of the rate decrease subsystem

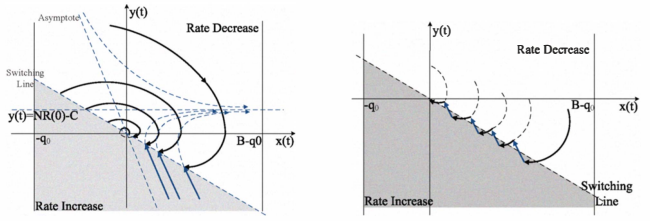
(b) Possible phase trajectories of the FR state

Fig. 3. The phase trajectories of QCN

Given an autonomous system described by differential equation $\ddot{x}(t) = f(x(t), \dot{x}(t))$, the trajectory of the system can be drawn on the phase plane by connecting points $(x(t), \dot{x}(t))$ along the direction where time t increases. The phase trajectory of second order differential equations has been presented by lots of literature [22]. There are also a wealth of methods to draw the phase trajectories of all kind of second order differential equations, such as the isocline, the liénard plane and the delta method [22]. Drawing phase trajectory is superior to displaying $x(t)$ and $y(t)$ against time t, since the latter one means finding analytical solution of differential equations through difficult deductions. Thus, we draw the phase trajectories of equation (11) on the phase plane here. Under practical system parameters, $\sqrt{\frac{G_d}{4C}}\frac{w}{p} < 1$, the phase trajectories of the rate decrease subsystem of QCN are shown in $Fig.3(a)$. The rate decrease subsystem itself is convergent. However, equation (11) holds only when $F_b(t) < 0$. With the increase of time t, $F_b(t)$ will become positive, namely the phase trajectories will "pierce" the switching line $F_b(t) = 0$. When $F_b(t)$ becomes positive, CP will not generate the feedback packet any more, and QCN enters into the FR state. In the same way, we can draw the phase trajectories of the process of FR and the process of AI. A branch of the phase trajectories of the process of FR at the condition of $NR(0) - C > 0$ are shown in $Fig.3(b)$.

Combining the phase trajectories of the rate decrease subsystem and the rate increase subsystem when $NR(0) - C > 0$, we find a special motion pattern of the QCN system. As shown in $Fig.4(a)$, when the QCN system is in the rate decrease area, the behavior of the phase trajectory is the same as the one shown in $Fig.3(a)$. Subsequently, the phase trajectory will "pierce" the switching line, entering into the rate increase area. In the rate increase area, the behavior of the phase trajectory is the same as the one shown in $Fig.3(b)$. The phase trajectory will "pierce" the switching line again immediately, and return back to the rate decrease area. As a result, QCN repeats this special motion pattern again and again, namely switching frequently between the FR state and the RD state. In details, the phase trajectory of the QCN system becomes the same as the solid line shown in $Fig.4(b)$. Finally, the QCN system approaches to equilibrium point along the switching line.

On average, or if the frequency of the switching between the FR state and RD state is infinite ideally, the system behavior



(a) Sliding mode motion of QCN (b) Sliding mode motion in details

Fig. 4. Sliding mode motion

will be equivalent to the one described by $F_b(t) = 0$, namely,

$$x(t) + \frac{w}{pC} y(t) = 0 \tag{12}$$

This motion pattern is called sliding mode motion [9]. The sliding mode motion has two advantages. The first one is that the second order QCN system is equivalent to the first order system defined by equation (12). The second advantage is that, when the system hits the sliding regime, namely the system enters into the sliding mode motion, the system behaviors become insensitive to system parameters and external disturbances. The changes of both traffic conditions and network states can be regarded as external disturbances of the QCN system.

When the phase trajectory of QCN reaches the switching line at the forth quadrant, the necessary and sufficient condition for the occurrence of the sliding mode motion is that $\lim_{F_b(t)\to 0^+} \dot{F}_b(t) \leq 0$ and $\lim_{F_b(t)\to 0^-} \dot{F}_b(t) \geq 0$ [9]. Inequality $\lim_{F_b(t)\to 0^-} \dot{F}_b(t) \geq 0$ means that once QCN enters into the rate decrease area in the forth quadrant, it moves back to the switching line and enters into state of FR immediately. Obviously, $\lim_{F_b(t)\to 0^-} \dot{F}_b(t) \geq 0$ always holds in the forth quadrant when $\sqrt{\frac{G_d}{4C}}\frac{w}{p} < 1$ in reality. Inequality $\lim_{F_b(t)\to 0^+} \dot{F}_b(t) \leq 0$ means that once QCN enters into the process of FR, it moves back to the switching line and enters into rate decrease area immediately. Referring equation (7) and (8), we can work out that $\lim_{F_b(t)\to 0^+} \dot{F}_b(t) \leq 0$ is equivalent to

$$(2T - \frac{w}{pC})x_0 \leq \frac{w^2}{p^2C^2}[NR(0) - C] \tag{13}$$

at the point (x_0, y_0) in the switching line. In inequality (13), T, w and p are system parameters. Parameters C and N are related to the network configuration. And x_0 and $R(0)$ reflects the current state of the QCN system. Thus, only under certain system state and configuration, the QCN system can enter into the sliding mode motion.

Inequality (13) also indicates that when x_0 is small, namely when the offset of the queue length to the target point is small, the QCN system is probably to enter into the sliding mode motion. After entering into the sliding mode motion, the QCN system approaches to the equilibrium point directly along the switching line, and accordingly the queue at the bottleneck link chatters around the target point. However, inequality (13) may not be satisfied, e.g., when T is large. In this condition, the QCN system fails to enter into the sliding mode motion and the queue at the bottleneck link oscillates. These analysis results consist with the above experiment results.

In a word, although QCN may move to the equilibrium point via the sliding mode motion, the heuristical rate adjustment algorithm can't guarantee that the QCN system enters into the sliding mode motion in any conditions. In this work, we intend to deliberately design an elaborate control structure to guarantee that the CM system can enter into the sliding mode motion under any conditions so as to provide a stable rate regulation algorithm, which is insensitive to system parameters and network configurations.

IV. The SMCC Scheme

A. Design Principles

The design principles of the SMCC scheme are as follows.

- SMCC should be able to hit the sliding regime starting from any initial states under any network configurations to utilize the advantages of the sliding mode motion that the system behaviors are insensitive to system parameters and external disturbances.
- SMCC should stick to the framework of the CM scheme developed by the IEEE 802.1 Qau work group. Because this framework integrates experiences of existing CM mechanisms and has been tested for several years. In this way, it is also convenient to replace QCN by SMCC.
- To handle traffics at the speed of $1Gbps$ or $10Gbps$, SMCC should be simple enough for hardware implementation. QCN is relatively complex.

– At RP, QCN needs to "remember" the current state for rate adjustment.
– In the process of FR and AI, the complex combination of timer and counter is used to identify time length of cycles and whether R_{HAI} should be used.
– Other add-on mechanisms, such as Extra Fast Recovery and Target Rate Reduction, are used to restore the performance of QCN in special environment.

- In data centers, the traffic is bursty and controllable low delay is required by HPC traffic. Thus, the response time of SMCC is expected to be as short as possible.

B. The SMCC Scheme

SMCC follows the framework developed by the IEEE 802.1 Qau work group. In brief, the differences between SMCC and QCN are

- At CP, the queue length and its derivative are involved in the feedback packet separately in SMCC. While the CP of QCN computes a weighted sum of them and embeds this sum into the feedback packet.
- At RP, SMCC use feedback information for rate increase and needs to identify the congestion point as BCN, instead of self-increase as QCN.
- The rate adjustment algorithm of SMCC differs from QCN.

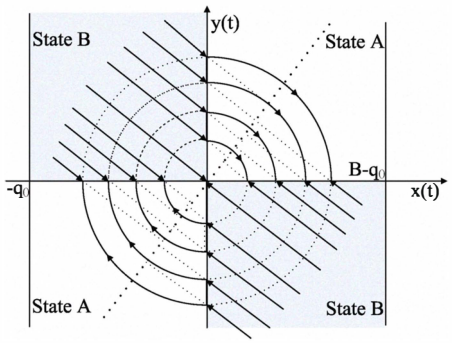


Fig. 5. The trajectory of SMCC

In SMCC, the feedback information involved in the feedback packet are Q_{off} and ΔQ, where Q_{off} is the offset of the queue length to the target point and ΔQ presents the variance of the queue length in a sampling interval. QCN simply compute a weighted sum of Q_{off} and ΔQ and feed this sum back. In this way, they lose valuable information. For example, QCN can't identify the rate decrease is mainly caused by long queue length, i.e., large Q_{off} or by quick variance of the queue length, i.e., large ΔQ.

SMCC uses the feedback information for both rate increase and rate decrease. Feedback information makes it possible to set proper sending rate at once, while a search algorithm, such as the self-increase algorithm used by QCN, at least needs to try several times for the proper sending rate. Thus, the rate adjustment algorithm of SMCC can be more precise and rapid. Thus, SMCC is expected to have shorter response time than QCN.

The rate adjustment algorithm of SMMC is as follows. The SMCC system is divided into two states as shown in $Fig.5$. Axis $x(t)$ and $y(t)$ are defined by equation (9). State A is that Q_{off} and ΔQ have the same signal. And State B is that Q_{off} and ΔQ have different signals. These states are differentiated by the feedback information, namely they need not to be "remembered" as in QCN. The linear rate adjustment algorithms for states of SMCC are

$$r \leftarrow \begin{cases} r - a * Q_{off} & in \quad State \quad A \\ r - b * \Delta Q & in \quad State \quad B \end{cases} \tag{14}$$

where a and b are positive constant parameters. Since the goal of the CM scheme is to hold the queue length around the target point, SMCC focuses on the degree that the current queue length deviates from the equilibrium point, instead of whether the sending rate is increased or decreased. In State A, the queue length is offset to the equilibrium point and will move away from the equilibrium point. Thus, short response time is required and the sending rate should be adjusted in large scale in State A. Let State A stands alone can help the system to achieve these goals. Differentiating the system states in this way, SMCC has two switching lines $x(t) = 0$ and $y(t) = 0$. In the following subsection, we will show that SMCC can hit the sliding regime $y(t) = 0$ for any $a, b > 0$, staring from any system state under any network configurations.

C. Theoretical Analysis

In this subsection, the theoretical foundations for designing SMCC are presented. To exhibit the procedure we design SMCC, we present the sufficient condition that the following general system hits the sliding regime $y(t) = 0$ staring from any state under any network configurations.

- The system is divided into State A and State B.
- The subsystems are linear, and the negative feedback is employed.

With the same method used in Section III, the general system can be modeled by the following differential equations. When $x(t)$ and $y(t)$ have the same signal,

$$\begin{cases} \frac{dx(t)}{dt} = y(t) \\ \frac{dy(t)}{dt} = -a_1(t)x(t) - a_2(t)y(t) \end{cases} \tag{15}$$

and when $x(t)$ and $y(t)$ have different signals,

$$\begin{cases} \frac{dx(t)}{dt} = y(t) \\ \frac{dy(t)}{dt} = -b_1(t)x(t) - b_2(t)y(t) \end{cases} \tag{16}$$

where $a_1(t), a_2(t), b_1(t), b_2(t)$ are nonnegative coefficients. These coefficients are associated with time t because parameters such as link capacity C, the number of flows N sharing the bottleneck link change with the time t.

We should firstly make sure that $y(t) = 0$ is the sliding regime, through which the system reaches the equilibrium point. Secondly, we should ensure the system hit the sliding regime starting from any state under any network configurations. The necessary and sufficient conditions for the switching line $y(t) = 0$ being the sliding regime is that the following inequalities hold [9]

$$\begin{cases} \lim\limits_{y(t)\to 0^+} \dot{y}(t) < 0 \quad when \quad x(t) > 0 \\ \lim\limits_{y(t)\to 0^+} \dot{y}(t) < 0 \quad when \quad x(t) < 0 \\ \lim\limits_{y(t)\to 0^-} \dot{y}(t) \geq 0 \quad when \quad x(t) \leq 0 \\ \lim\limits_{y(t)\to 0^-} \dot{y}(t) \geq 0 \quad when \quad x(t) \geq 0 \end{cases} \tag{17}$$

Inequalities (17) means the phase trajectory of the system will "pierce" the line $y(t) = 0$ from both sides in the neighborhood of the line $y(t) = 0$. Referring to (15) and (16), inequalities (17) can be simplified as

$$\begin{cases} -a_1(t)x(t) < 0 \quad when \quad x(t) > 0 \\ -b_1(t)x(t) < 0 \quad when \quad x(t) < 0 \\ -a_1(t)x(t) \geq 0 \quad when \quad x(t) \leq 0 \\ -b_1(t)x(t) \geq 0 \quad when \quad x(t) \geq 0 \end{cases} \tag{18}$$

Therefore, the necessary and sufficient condition for the inequalities (17) always holds is

$$\min\{a_1(t)\} > 0 \quad and \quad \max\{b_1(t)\} \leq 0. \tag{19}$$

Once inequality (19) holds, $y(t) = 0$ is the sliding regime of the whole system.

The next step is to explore the sufficient condition for that the system can hit the sliding regime under any network

configurations. When inequality (19) holds, there must be $b_1(t) = 0$ since $b_1(t)$ is nonnegative. Therefore, the trajectories of the system described by (15) and (16) are straight lines in the second and forth quadrant as shown in $Fig.5$. Since both $a_1(t)$ and $a_2(t)$ are nonnegative, the characteristic equation $p^2 + a_2(t)p + a_1(t) = 0$ have no nonnegative roots for any t. Consequently, the trajectories of the system described by (15) and (16) are cycles or ellipses in the first and third quadrant for any t. Totally, the trajectories of the system are cycles or ellipses in the first and third quadrant. Therefore, the system described by (15) and (16) can hit the sliding regime starting from any initial states under any network configurations.

As a special case of the general system, SMCC chooses parameters $a = a_1(t) > 0$, $a_2(t) = 0$, $b_1(t) = 0$ and $b = \frac{b_2(t)}{pC} > 0$, referring to equations (14), (15) and (16). In this way, the trajectories of the SMCC system are as the solid line shown in $Fig.5$. Naturally, the SMCC system can hit the sliding regime $y(t) = 0$ under any network configurations. Once entering into the sliding mode motion, SMCC approaches to the equilibrium point along $y(t) = 0$.

D. Parameters Settings

Although the sliding mode motion is insensitive to the system parameters theoretically, the granularity of the rate adjustment should be considered in practice. Thus parameters a and b should be chosen carefully in SMCC. Theoretically, large a and b are expected for shorter response time. But in reality, the granularity of the rate adjustment, which is determined by the values of a and b obviously, should be considered in SMCC. Naturally, large a and b are required for large bandwidth, and so forth. Unfortunately, the available bandwidth shared by RPs is time-varying. Hence, parameters in equation (15) and (16) are required to change with the bandwidth, namely parameters $a_1(t)$ and $b_2(t)$ are required to change with the bandwidth in SMCC.

In SMCC, parameter $b_2(t) = bpC$ is already associated with the bandwidth when b is constant. Hence, there is no need to let parameter b change with time t. Similar conclusion can also be obtained from equation (14), in which ΔQ is already associated with the bandwidth. On the other hand, the changes of bandwidth can be inferred by RPs in SMCC through monitoring the speed of incoming feedback packets or the feedback information. Therefore, it is possible let parameter a change with the bandwidth. However, this method will increase the complexity of SMCC.

We realize that short response time is needed only when the traffic is excessive, and thus present a two stages way to set parameter a. Parameter a is normally set a small constant value a_{small}. The feedback information ΔQ and Q_{off} can be used as the indicator of traffic variance. Thresholds T_1 and T_2 are defined for ΔQ and Q_{off}, respectively. Then, in State A

- When $|\Delta Q| > T_1$ and $|Q_{off}| > T_2$, parameter a is set a large constant value a_{large}.
- Or else, parameter a turns back to the small constant value a_{small}.

In this way, the SMCC system can response rapidly when the traffics is excessive, and achieve fine-grained granularity of the rate adjustment in the other conditions.

E. Discussions

For the convenience of discussion, we define four states of the CM system: State (1) $Q_{off} > 0$ and $Nr(t) < C$, State (2) $Q_{off} < 0$ and $Nr(t) < C$, State (3) $Q_{off} < 0$ and $Nr(t) > C$, State (4) $Q_{off} > 0$ and $Nr(t) > C$. Naturally, the queue length keeps decreasing until $Nr(t) \geq C$ in State (1). If the sending rate is increased slowly as in BCN, Q_{off} becomes small than zero before $Nr(t) \geq C$. Subsequently, the BCN system will enter into State (2). In this condition, the queue length (Q_{off}) oscillates severely and the response time is large. Therefore, BCN is unsuitable for the burst traffic in data centers, which is the reason that BCN was replaced by QCN historically. The QCN system employs the fast recovery algorithm for State (1). Hence, the speed of the sending rate increases is decided by the target rate for fast recovery in State (1). When the target rate is large enough, the sending rate is increased fast enough. The system will enter into the sliding mode motion. Or else, QCN experiences the same sequence of the system states as BCN. Thus, the performance of QCN depends on the system state. In contrast, the SMCC system will enter into the sliding mode motion after State (1) and then switch either between State (1) and State (2) or between State (1) and State (3) as shown in $Fig.5$. Finally, the SMCC system approaches to the equilibrium point directly.

V. Implementation and Evaluation

A. Implementation

NetFPGA [11] is a programmable hardware platform for fast prototyping. It consists of a Xilinx Virtex-II Pro FPGA, four $1Gbps$ Ethernet ports and 4MB SRAM. To verify our theoretical analysis, we implement QCN and SMCCC on the NetFPGA platform. The CP of QCN and SMCC are almost the same, except for the format of the feedback packets. It is implemented by extending the Reference Switch project [11]. The RP is implemented by extending the Packet Generator project [11]. The difference in RP between QCN and SMCC is the rate adjustment algorithm, namely a calculator on the NetFPGA platform. The calculator of QCN is more complex than that of SMCC since QCN need to "remember" states.

B. Experiment Setup

Our experiments are conducted on two network scenarios. Scenario I uses the 3-sources dumbbell topology. In scenario II, the parking lot topology is configured. As shown in $Fig.6$, C1, C2 and C3 are CPs. S1, S2 are RPs, and their workloads are long-lived flows destined to R1 and R2, respectively. S3 starts a flow of size $750Mbps$ destined to R3 at the 1st second. At the 3rd second, the flow from S3 stops, but S4 starts a flow of size $875Mbps$ destined to R4, which stops at the 5th second. RPs start long-lived flows at the speed of $1Gbps$. Note that the bottleneck is at C1 in the first 3 seconds and at C2 in the last 2 seconds. The parameters setting of QCN is the same as

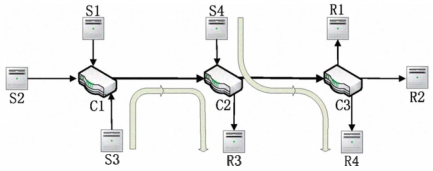
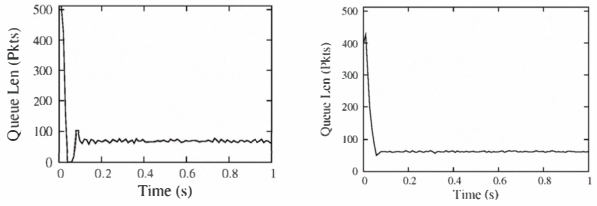


Fig. 6. Parking Lot Topology



(a) QCN with T set to be the time of sending $15kB$ data (b) SMCC with $R_a = 256$ and $R_b = 256$

Fig. 7. Evolution of the queue length under scenario I

that in Section III. Parameters a and b in SMCC are chosen such that the maximum adjustment range of the sending rate is R_aMbps and R_bMbps in a sampling interval, respectively. These parameters keeps unchanged unless declared explicitly.

C. Evaluation

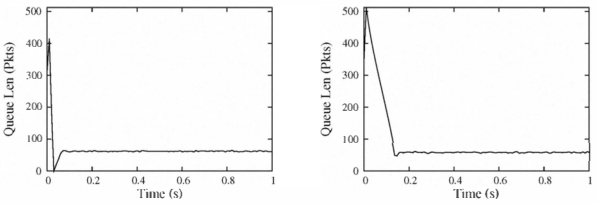
The evolution of the queue length at the bottleneck link reflect the performance of the CM schemes. Two metrics are used for the performance evaluation.

- The response time, namely the time the CM scheme takes to move to the equilibrium point after the changes of system parameters or network configurations.
- The stability of the CM system. Stable CM system means the queue at the bottleneck link is neither emptied nor overflowed.

1) Comparing QCN with SMCC: The evolution of the queue length at the bottleneck link under scenario I are as shown in $Fig.7$. In QCN, parameter T is set to be the time of sending $15KB$ data. And in SMCC, $R_a = 256$ and $R_b = 256$. $Fig.7$ shows that SMCC has smaller response time than QCN, and is stable. Both QCN and SMCC have stable queue except for the regulation process.

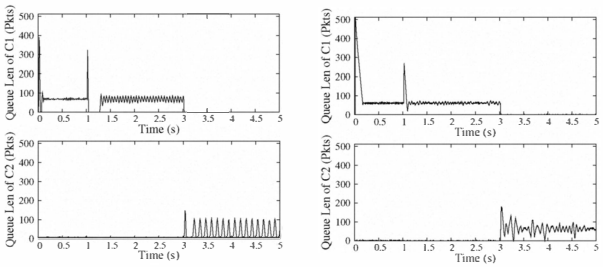
In Section III, we have shown that QCN becomes unstable with the change of parameter T as shown in $Fig.2$. By contrast, experiments verify that SMCC is insensitive to system parameters. For example, when parameters are reset by ($R_a = 256$, $R_b = 32$) or ($R_a = 128$, $R_b = 256$) under scenario I, SMCC keeps stable except for the regulation process, as shown in $Fig.8$. The main behaviors and motion pattern of SMCC are unchangeable with the change of parameters.

Similar experiments are conducted under scenario II. In QCN, parameter T is reset to be the time of sending $15KB$ data and $R_a = 256$, $R_b = 256$ in SMCC. The evolution of the queue length at the bottleneck link are shown in $Fig.9$. Experimental results show that SMCC keeps stable with the changes of bandwidth, while QCN becomes unstable after the bandwidth is preempted by $875Mb$ after the 3rd second.



(a) $R_a = 256$ and $R_b = 32$ (b) $R_a = 128$ and $R_b = 256$

Fig. 8. Evolution of the queue length under scenario I in SMCC



(a) QCN with T set to be the time of sending $15KB$ data (b) SMCC with $R_a = 256$ and $R_b = 256$

Fig. 9. Evolution of the queue length under scenario II

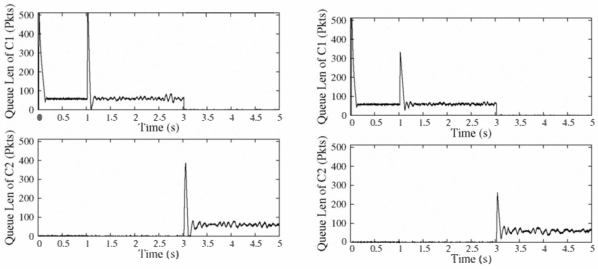
Moveover, QCN can't response to the changes of bandwidth timely at the 1st second and at the 3rd second such that the queue is emptied for a long time. Thus, SMCC is superior to QCN in the time-varying network environment.

In a word, SMCC has shorter response time than QCN, and is stable. Moreover, SMCC is insensitive to system parameters and network configurations.

2) Impacts of Parameters in SMCC: Though SMCC keeps stable with the changes of system parameters, the parameters setting decides the granularity of the rate adjustment in SMCC and thus influence its performance. Comparing $Fig.7(b)$ with $Fig.8$, we can find that the parameters influence the response time of the SMCC system. More specially, parameter a dominates the response time, and parameter b can help to inhibit the empty buffer caused by burst traffic.

Although the bottleneck link queue system in SMCC keep stable in $Fig.9(b)$, the queue oscillates after the 3rd second. The main reason is that parameter a is too large. We should note that R_a is larger than the bandwidth shared by RPs, namely $125Mbps$ after the 3rd second. In this condition, the rate adjustment algorithm is coarse. When R_a is reduced to $128Mbps$, the queue only chatters between the 3rd second and 5th second as shown in $Fig.10(a)$. Fine-gained granularity of the rate adjustment algorithm is obtained. But the sending rate can't be adjusted timely such that the queue is emptied when the traffic changes quickly at the 1st second and at the 3rd second. To solve this problem, we try the two stages way to set parameters.

When the traffic changes quickly, parameter a is set a large value, namely the maximum adjustment range of the sending rate is R_{large}. When the traffic changes slowly, parameter a is set a small value, namely the maximum adjustment range of the sending rate is R_{small}. The speed of the traffic changes can



(a) $R_a = 128$ and $R_b = 256$ (b) $R_{large} = 256$, $R_{small} = 128$ and $R_b = 32$

Fig. 10. Evolution of the queue length under scenario II

be inferred from the feedback information. In our experiment, $Q_{off} > 16KB$ and $\Delta Q > 1KB$ are considered as the threshold to detect the traffic with rapid change. The evolution of the queue length at the bottleneck link under this parameter setting are shown in $Fig.10(b)$. In the condition of small bandwidth, namely after the 3rd second, the evolution of the queue length is also smooth. And moreover, the sending rate is adjusted timely such that the queue is hardly emptied. The performance of SMCC is improved by using the two stages way to set parameters.

According to our experiments, the recommended parameters setting for $1Gbps$ links are $T_1 = 1KB$, $T_2 = 16KB$ and choosing a_{large}, a_{small} and b such that the maximum adjustment range of sending rate is $256Mbps$, $128Mbps$ and $256Mbps$, respectively.

3) Remarks: Mathematically, ΔQ is the function of the derivative of Q_{off}. The variance of ΔQ is small comparing to that of Q_{off}. Referring to $Fig.10(a)$, R_b is larger than the bandwidth sharing by RPs after the 3rd second in the parking lot topology, but R_b is hardly reached. In fact, ΔQ keeps a small value throughout our experiments. That's why the threshold $T_1 = 1KB$ of ΔQ is set a smaller value comparing to the threshold $T_2 = 16KB$ of Q_{off}. Thus, it's safe to set parameter b a large value in reality. Knowing that the variance of Q_{off} is larger than that of ΔQ, we can infer that parameter a dominates the granularity of the rate adjustment. Referring that parameter a also dominates the response time, we can explain why parameter a should not be set too large but in a two stages way.

VI. Conclusion

As the Internet owns its scalability and stability to TCP, the CM scheme would also play a central role in DCE networks. Though QCN has been ratified to be the standard for the CM scheme, experiments and theoretical analysis show that QCN may become unstable in certain conditions. With the insights that QCN reaches the stable state mainly via the sliding mode motion, the SMCC scheme is guaranteed to enter into the sliding mode motion under any conditions. Theoretically, SMCC benefits from the advantage that the sliding mode motion is insensitive to system parameters and network configurations. Therefore, SMCC is stable and robust. Moreover, SMCC is simpler and can achieve smaller response time than QCN. Experiments on the NetFPGA platform verify these results. In term of the deployment, SMCC follows the basic CM framework defined by the IEEE 802.1 Qau work group, and thus QCN can be replaced by SMCC through modifying only the firmware.

VII. Acknowledgement

The authors gratefully acknowledge the anonymous reviewers for their constructive comments. This work is supported in part by the National Natural Science Foundation of China (NSFC)under Grant No. 60971102 and 60932003, National Basic Research Program of China (973 Program) under Grant No. 2009CB320504 and 2012CB315803, and National Science and Technology Major Project of China (NSTMP) under Grant No.2011ZX03002-002-02.

References

[1] *IEEE 802.1: Data Center Bridging Task Group*, http://www.ieee802.org/1/pages/dcbridges.html
[2] *Unified Fabric: Csico's Innovation for Data Center Networks*, white paper, 2008, http://www.cisco.com/en/US/solutions/collateral/ns340/ns517/ns224/ns783/white_paper_c11-462422.pdf
[3] M. AI-Fares,A. Loukissas and A. Vahdat, *A Scalable, Commodity Data Center Network Architecture*,in Proc. of the ACM SIGCOMM, 2008.
[4] S. Kandula, S. Sengupta, A. Greenberg, and P. Patel, *The nature of datacenter traffic: Measurements and Analysis*, in Proc. of the ICM, 2009.
[5] C. DeSanti and J. Jiang, *FCoE in Perspective*, in Proc. of the International Conference On Advanced Infocomm Technology, 2008.
[6] D. Cohen, T. Talpey, A. Kanevsky, U. Cummings, M. Krause, R. Recio, D. Crupnicoff, L. Dickman and P. Grun, *Remote Direct Memory Access over the Converged Enhanced Ethernet Fabric: Evaluating the Options*, in Proc. of the High Performance Interconnects, 2009.
[7] B. Chia, *DC Technology Update*, http://www.cisco.com/web/SG/learning/dc_partner/files/Nexus_Update.pdf, 2010.
[8] *Fulcrum Announces 1 Billion Packet Per Second 10G/40G Ethernet Switch Chips for Efficient Scaling of Virtualized Data Center Networks*, http://www.granitevc.com/files/news/2010/node-447/Anncmt_10-1102.pdf, Nov. 2010.
[9] U. Itkis, *Control systems of variable structure*, by Keter Publishing House Jerusalem Ltd. 1976
[10] L. Xu, K. Harfoush and I. Rhee, *Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks*, in Proc. of the INFOCOM, 2004.
[11] *The NetFPGA Projects*, http://netfpga.org/foswiki/bin/view/NetFPGA/OneGig/ProjectTable.
[12] V. Misra and W.B. Gong and D. Towsley, *Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED*, in Proc. of the ACM SIGCOMM, Aug. 2000.
[13] V. Jacobson, *Congestion Avoidance and Control*, in Proc. of the ACM SIGCOMM, 1988.
[14] T. Kelly, *Scalable TCP: improving performance in highspeed wide area networks*, in Proc. of the ACM SIGCOMM CCR, 2003.
[15] *IEEE 802.1Qau: End-to-end congestion management, Working Draft*,http://www.ieee802.org/1/pages/802.1au.html
[16] D. Bergamasco and R. Pan, *Backward Congestion Notification Version 2.0*, IEEE 802.1 Meeting, September 2005
[17] J. Jiang, R. Jain and C. So-In, *An Explicit Rate Control Framework for Lossless Ethernet Operation*, in Proc. of the IEEE ICC, 2008.
[18] M. Gusat, *Extended Ethernet Congestion Management(E2CM)*, May. 2007, http://www.ieee802.org/1/files/public/docs2007/
[19] R. Pan and B. Prabhakar and A. Laxmikantha, *QCN: Quantized Congestion Notification*, May. 2008, http://www.ieee802.org/1/files/public/docs2008/
[20] A. Kabbani and M. Yasuda, *Data Center Quantized Congestion Notification (QCN): Implementation and Evaluation on NetFPGA*, the 1st Asia NetFPGA Developers Workshop, Daejeon, Korea, June 14, 2010.
[21] E.A. Coddington and N. Levionson, *Theroy of Ordinary Differential Equations*, McGraw Hill, 1975.
[22] Derek P. Atherton, *Nonlinear Control Engineering*, Van Nostrand Reinhold Company, 1982.