Fair Quantized Congestion Notification in Data Center Networks

---

Citation:
Yan Zhang and Nirwan Ansari, "Fair Quantized Congestion Notification in Data Center Networks," *IEEE Transactions on Communications*, DOI: 10.1109/TCOMM.2013.102313.120809, vol. 61, no.11, pp. 4690-4699, Nov. 2013

URL:
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6648356

Fair Quantized Congestion Notification in Data Center Networks

Yan Zhang, *Student Member, IEEE,* and Nirwan Ansari, *Fellow, IEEE*

***Abstract*—Quantized Congestion Notification (QCN) has been developed for IEEE 802.1Qau to provide congestion control at the Ethernet Layer or Layer 2 in data center networks (DCNs) by the IEEE Data Center Bridging Task Group. One drawback of QCN is the rate unfairness of different flows when sharing one bottleneck link. In this paper, we propose an enhanced QCN congestion notification algorithm, called fair QCN (FQCN), to improve rate allocation fairness of multiple flows sharing one bottleneck link in DCNs. FQCN identifies congestion culprits through joint queue and per flow monitoring, feedbacks individual congestion information to each culprit through multicasting, and ensures convergence to statistical fairness. We analyze the stability and fairness of FQCN via Lyapunov functions and evaluate the performance of FQCN through simulations in terms of the queue length stability, link throughput and rate allocations to traffic flows with different traffic dynamics under three network topologies. Simulation results confirm the rate allocation unfairness of QCN, and validate that FQCN maintains the queue length stability, successfully allocates the fair share rate to each traffic source sharing the link capacity, and enhances TCP throughput performance in the TCP Incast setting.**

***Index Terms*—Data Center Network (DCN), Quantized Congestion Notification (QCN), congestion notification.**

I. Introduction

Data centers, typically composed of storage devices, servers and switches for interconnecting servers, are becoming increasingly important to provide a myriad of services and applications to store, access, and process data. Owing to the inherent merits of Ethernet, such as low cost, ubiquitous connectivity, and ease of management, Ethernet has become the primary network protocol to provide a consolidated network solution for data center networks (DCNs). However, Ethernet was originally designed for best-effort communications in a local area network (LAN) and not optimized for DCNs. The Data Center Bridging Task Group [1] in the IEEE 802.1 Ethernet standards body thus aims to enhance classical switched Ethernet to provide more services for DCNs. Project IEEE 802.1Qau is concerned with specifications of the Ethernet Layer or Layer 2 congestion notification mechanism for DCNs.

Several congestion notification algorithms have been proposed and developed to reduce or eliminate packet drops at the congestion switch in DCNs, e.g., Backward Congestion Notification (BCN) [2, 3], Forward Explicit Congestion Notification (FECN) [4], the enhanced FECN (E-FECN) [5], and Quantized Congestion Notification (QCN) [6–8]. It has been shown that BCN achieves only proportional fairness but not max-min fairness [9]. FECN and E-FECN can achieve perfect fairness, but the control message overhead is high. QCN can effectively control link rates to resolve switch congestions in several round trips, i.e., several hundred micro seconds with 10 Gbps link capacity in today's typical DCNs. However, one drawback of QCN is the rate unfairness of different flows when sharing one bottleneck link. Such rate unfairness also degrades the TCP throughput in synchronized readings of data blocks across multiple servers [10]. We thus propose an enhanced QCN congestion control algorithm, called fair Quantized Congestion Notification (FQCN) [11], to improve fairness of multiple flows sharing one bottleneck link. The main idea of FQCN is to distinguish congestion culprits and feedback the global and per-flow congestion information to all culprits when the switch is congested. In this paper, we extend the original FQCN algorithm [11] by introducing the weighted rate allocation. We also provide theoretical analysis on the stability and fairness of the proposed FQCN algorithm via Lyapunov functions. Moreover, we conduct more simulations to evaluate the performance of FQCN in terms of the queue stability, link throughput and rate allocations to traffic flows with different traffic dynamics under three network topologies.

The rest of the paper is organized as follows. In Section II, we summarize the related works. In Section III, we overview the QCN algorithm. Then, we describe the proposed FQCN congestion notification mechanism and analyze FQCN analytically via Lyapunov method in terms of stability and fairness in Sections IV and V, respectively. The performance evaluation of the proposed FQCN is provided in Section VI. Finally, Section VII concludes the paper.

II. Related Works

BCN [2, 3], FECN [4], E-FECN [5] and QCN [6–8] are all queue sampling based congestion notification algorithms. They all assume that the switches detect congestion by sampling the queue and generate the feedback congestion notification messages to the sources based on the calculated congestion parameter. The rate regulators at the sources adjust the rates of individual flows according to congestion notification messages received from the switches.

Both BCN [2, 3] and QCN [6–8] are rate-based closed-loop feedback control mechanisms between the rate regulator located at a source and a switch. In BCN, a switch feeds back not only negative information regarding congestion but also positive rate increasing information, and the rate regulator in BCN adjusts its rate by using a modified Additive Increase and Multiplicative Decrease algorithm. In QCN, only negative congestion feedback is used. Thus, the control overhead of QCN is

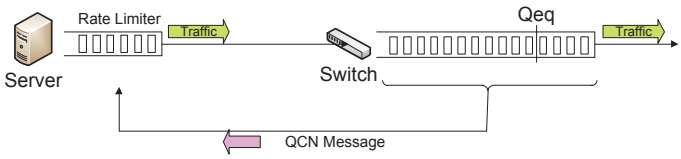
Manuscript received Oct. 25, 2012; revised May 19 and Sep. 1, 2013.

Y. Zhang and N. Ansari are with the Advanced Networking Lab., Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, 07102 USA. E-mail: {yz45, nirwan.ansari}@njit.edu.

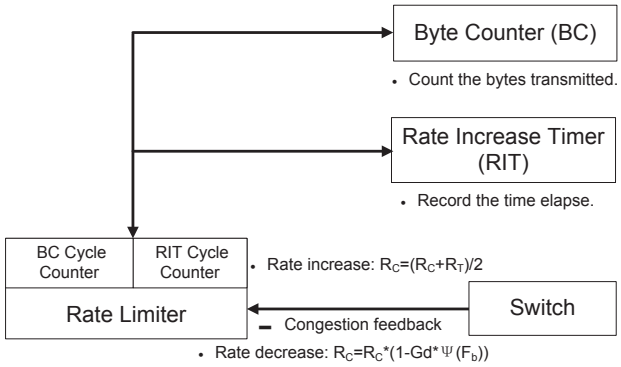
smaller than that of BCN. Since there is no positive feedback in QCN, the rate regulators in QCN provide mechanisms to recover the lost bandwidth and probe for extra available bandwidth voluntarily. Moreover, the sampling probability is constant in BCN while it is proportional to the congestion parameter in QCN.

To ensure scalability, one of the main requirements and design rationales of IEEE 802.1 for QCN is to enforce the QCN enabled switch to be "no state", i.e., not storing any per flow state. This requirement also prevents QCN from providing any fairness stronger than proportional fairness, such as max-min fairness. An enhancement to QCN, called approximate fairness QCN (AF-QCN) [12], was proposed to improve the fairness allocation of link capacity among all sharing sources. AF-QCN extends the functionality of QCN with an approximate fairness controller to provide weighted fairness on a per-flow or per-class basis. For each incoming flow, the Ethernet switch with AF-QCN estimates its arrival rate, calculates its fair share, and derives its dropping probability.

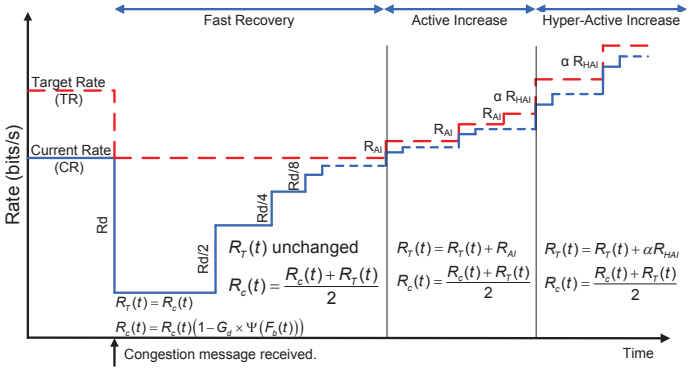
FECN [4] is a close-loop end-to-end explicit rate feedback control mechanism. Each source periodically probes the congestion condition along the path to the destination. The rate field in the probe message is modified along the forward path by the switch if the available bandwidth at the switch in the forward path is smaller than the value of the rate field in the probe message. When a source receives the probe message returned back from the destination, the rate regulator adjusts the sending rate as indicated in the received message. All flows are treated fairly in FECN since the same rate is advertised by the switch. E-FECN [5] enhances FECN by allowing the switches feedback to the source directly under severe congestion. Reference [13] provides a more detailed discussion on the Ethernet layer congestion notification algorithms for DCNs.



(a) QCN system model.



(b) QCN control mechanism.



(c) QCN rate limiter operation.

Fig. 1: QCN overview

III. Quantized Congestion Notification

The QCN algorithm is composed of two parts as shown in Fig. 1(a): switch or congestion point (CP) dynamics and rate limiter (RL) dynamics. At CP, the switch buffer attached to an oversubscribed link samples incoming packets and feeds the congestion severity level back to the source of the sampled packet. At one traffic source, RL decreases its sending rate based on the congestion notification message received from CP, and increases its rate voluntarily to recover lost bandwidth and probe for extra available bandwidth. The rate control mechanism and the operations of RL in QCN are summarized in Fig. 1(b) and (c), respectively. In order to ease the understanding, a list of all symbols used in the paper is summarized in Table I.

A. The CP Algorithm

The goal of CP is to maintain the buffer occupancy at a desired operating point, Q_{eq}, which is the queue threshold for equilibrium. At time t, CP samples the incoming packets with a sampling probability $p(t)$, and computes the congestion feedback value $F_b(t)$. The sampling probability is initialized to 1%, and is updated after computing the congestion feedback value $F_b(t)$ at each sampling event. Denote $Q_{len}(t)$ and $Q_{len}(t-\tau)$ as the instantaneous queue length in bits of the current sampling event at time t and last sampling event at time $t-\tau$, respectively, where τ is the time interval between two adjacent sampling events. The congestion feedback value $F_b(t)$ consists of a weighted sum of the instantaneous queue offset $Q_{over}(t)=Q_{len}(t)-Q_{eq}$ and the queue variation over the last sampling interval $Q_\delta(t)=Q_{len}(t)-Q_{len}(t-\tau)$, as defined below:

$$F_b(t)=-(Q_{over}(t)+w*Q_\delta(t)) \tag{1}$$

where w is a non-negative constant, taken to be 2 for the baseline implementation. In fact, $Q_{over}(t)$ indicates queue-size excess while $Q_\delta(t)$ indicates rate excess. If $F_b(t)$ is negative, either the buffer or the link or both are oversubscribed and a congestion notification message containing the value of quantized $|F_b(t)|$, denoted as $\Psi(F_b(t))$, is sent back to the source of the sampled packet; otherwise, no feedback message is sent.

At each sampling event, the sampling probability is updated

as a function of $F_b(t)$ as follows:

$$p(t) = \begin{cases} (1 + \frac{9}{64}\Psi(F_b(t)))\% & (F_b(t) < 0) \\ 1\% & (F_b(t) \geq 0) \end{cases} \quad (2)$$

From the above equation, we can see that if there is no congestion ($F_b(t) \geq 0$), CP checks the congestion status with a probability of 1%; otherwise, the sampling probability is increased as a linear function of the quantized congestion feedback value $\Psi(F_b(t))$. In the default implementation, the congestion feedback value $F_b(t)$ is quantized to 6 bits and the maximum quantized value of $F_b(t)$ is 64, and therefore, the maximum sampling probability is 10%.

B. The RL Algorithm

As shown in Fig. 1 (c), RL adjusts the sending rate of the associated traffic source by decreasing the sending rate based on the quantized congestion feedback value contained in the congestion notification message, and increasing the sending rate voluntarily to recover lost bandwidth and probe for extra available bandwidth.

1) Rate decrease: when a congestion notification message is received, the current sending rate $R_C(t)$ is set as the target rate $R_T(t)$ and the current rate is reduced by a factor of $R_d(t) = R_C(t)G_d \times \Psi(F_b(t))$ as follows:

$$\begin{aligned} R_T(t) &= R_C(t) \\ R_C(t) &= R_C(t)(1 - G_d \times \Psi(F_b(t))) \end{aligned} \quad (3)$$

where the constant G_d is chosen to ensure that the sending rate cannot decrease by more than 50%, and thus $G_d * \Psi(F_{bmax}) = \frac{1}{2}$, where F_{bmax} denotes the maximum of $F_b(t)$. In the default implementation, the maximum quantized value of $F_b(t)$ is 64; thus, G_d is configured to 1/128.

2) Rate increase: Two modules, Byte Counter (BC) and Rate Increase Timer (RIT), are introduced in RL for rate increases. BC is a counter for counting the number of bytes transmitted by the traffic source, which is used to increase the sending rate by RL. Based on BC only, it will take a long time for a low rate source to increase its sending rate; this might result in low bandwidth utilization if there is tremendous available bandwidth. Moreover, the unfair sharing of bottleneck bandwidth can be observed when a low rate flow competes for the bandwidth with a high rate flow. In order to enable fast bandwidth recovery, rather than getting stuck at a low sending rate for a long time, RIT is introduced in RL to increase the source's sending rate periodically instead of based on the amount of data transferred as BC does.

BC and RIT work in two phases, Fast Recover (FR) and Active Increase (AI). At the FR phase, BC counts data bytes transmitted and increases the BC cycle by 1 when B_L bytes are transmitted. After each cycle, RL increases its rate to recover some of the bandwidth it lost at the previous rate decrease episode. After the C_T cycle (where C_T is a constant chosen to be 5 cycles in the baseline implementation), BC enters the AI phase to probe for extra bandwidth on the path. In the AI phase, RL counts the data bytes transmitted and increases the BC cycle by 1 when $B_L/2$ bytes are transmitted. BC is reset every time a rate decrease is applied and enters the FR state.

RIT functions similarly as BC. In the FR phase, RIT completes one cycle with T ms duration. After it counts out C_T cycles of T ms duration, it enters the AI phase where each cycle is set to $T/2$ ms long. It is reset when a congestion notification message arrives and enters the FR phase.

RL works in the FR, AI or Hyper-Active Increase (HAI) phase depending on the state of BC and RIT. BC and RIT jointly determine rate increases of RL, when either BC or RIT completes one cycle of data transfer. When a rate increase event occurs, the update of the current rate $R_C(t)$ and target

TABLE I: **Description of Symbols**

Symbols	Description
Network and Traffic Parameters	
$\mathcal{L}$	The set of links.
C_l	The capacity of link l.
$\mathcal{S}$	The set of sources.
$\mathcal{S}_l$	The set of sources using link l.
$\mathcal{S}_l^L$	The set of low rate flows on link l.
$\mathcal{S}_l^H$	The set of high rate flows on link l.
$\mathcal{S}_l^R$	The set of overrated flows on link l.
$R_i(t)$	The rate of source i.
$\overrightarrow{R(t)}$	The vector of rates of all sources.
$a_{l,i}$	The fraction of traffic from source i over the link l.
$f_l(t)$	The total amount of traffic over link l.
Queue Parameters	
$Q_{len}(t)$	The instantaneous queue length at time t.
Q_{eq}	The queue threshold for equilibrium.
$Q_{over}(t)$	The excess queue length that exceeds the equilibrium Q_{eq} at time t.
Q_δ(t)	The queue variation over the last sampling interval.
QCN Parameters	
$F_b(t)$	The congestion feedback value calculated at time t.
$\Psi(F_b(t))$	The quantized congestion feedback value of $F_b(t)$.
w	The weight parameter for the queue growth rate $Q_\delta(t)$ in computing the congestion feedback value $F_b(t)$.
G_d	The multiplicative decrease parameter of rate.
$p(t)$	The time varying sampling probability with which the CP samples the incoming packets.
C_T	The cycle threshold to determine the state of byte counter or rate increase timer, in the FR or AI phase.
B_L	Bytes transmitted to complete one byte counter cycle.
T	The Rate Increase Timer period in milliseconds.
$R_C(t)$	The current sending rate.
$R_T(t)$	The target sending rate
R_{AI}	The sending rate incremental parameter when one cycle of Byte Counter or Rate Increase Timer completes in AI phase.
R_{HAI}	The sending rate incremental parameter when one cycle of Byte Counter or Rate Increase Timer completes in HAI phase.
FQCN Parameters	
$B_i(t)$	The byte counts of flow i at time t.
W_i	The weight coefficient for flow i.
$M_i(t)$	The fair share for flow i on link l at the sampling time t.
$M_i^F(t)$	The fine-grained fair share for high rate flow $i \in \mathcal{S}_l^H$ on link l at the sampling time t.
$\Psi_{F_b}^F(i,t)$	The quantized congestion feedback value calculated with FQCN for the overrated source i at time t.
Analysis Parameters	
$p_l(f_l(t))$	The probability of generating a negative QCN congestion notification message from link l with load $f_l(t)$ at time t.
$P_l(t)$	The cumulative probability of generating a negative QCN congestion notification message from link l at time t.
$P(\overrightarrow{R(t)})$	The cumulative probability of generating a negative QCN message from all links at time t with vector of rates $\overrightarrow{R(t)}$.
$J_A(\overrightarrow{R})$	The Lyapunov function of vector $\overrightarrow{R}$.

To ease the reading and analysis, bits and bits/s are used as the units of queue length and transmission rate, respectively.

rate $R_T(t)$ in different RL state is summarized as follows:
a) RL is in FR if both BC and RIT are in FR. In this case, when either BC or RIT completes a cycle, the target rate $R_T(t)$ remains unchanged while the current rate $R_C(t)$ is updated as:

$$R_C(t) = \frac{1}{2}(R_C(t)+R_T(t)) = R_c(t)+\frac{1}{2}(R_T(t)-R_c(t)) \quad (4)$$

b) RL is in AI if either BC or RIT is in AI. In this case, when either BC or RIT completes a cycle, the current rate and target rate are updated as:

$$\begin{array}{rcl} R_T(t) & = & R_T(t) + R_{AI} \\ R_C(t) & = & \frac{1}{2}(R_C(t) + R_T(t)) \end{array} \quad (5)$$

where R_{AI} is a constant sending rate increment for RL in the AI state, and it is set to be 5 Mbps in the baseline implementation.
c) RL is in HAI if both BC and RIT are in AI. In this case, the target rate and current rate are updated as:

$$\begin{array}{rcl} R_T(t) & = & R_T(t) + \alpha * R_{HAI} \\ R_C(t) & = & \frac{1}{2}(R_C(t) + R_T(t)) \end{array} \quad (6)$$

where R_{HAI} is a constant sending rate increment for RL in the HAI state, and is set to 50 Mbps in the baseline implementation, and α is the minimum cycle count of BC and RIT in the AI state.

IV. FQCN Algorithm

One drawback of QCN is the rate allocation unfairness of different flows when sharing one bottleneck link, which can be seen clearly in our simulation results shown in Section VI. To improve the fair allocation of link capacity among all sharing sources in QCN, the proposed fair Quantized Congestion Notification (FQCN) feeds the congestion notification messages back to all flow sources which send packets at rates above their fair shares of the bottleneck link capacity. FQCN differs from QCN at the CP algorithm. As we can see from Eq. (1), QCN randomly samples the traffic packets at CP and feedbacks a global congestion status sequentially to a single traffic source deemed as the congestion culprit. This random congestion feedback in QCN prevents the sending rates from converging to statistical fairness. FQCN addresses these deficiencies: 1) it identifies the overrated flows, which are deemed as congestion culprits and whose sending rates are larger than their fair share rates; 2) through joint queue and per flow monitoring, it feedbacks individual congestion status to each culprit through multi-casting, thus ensuring convergence to statistical fairness.

A. Congestion Culprits Identification

Consider a network consisting of a set $\mathcal{L} = \{1, \cdots, L\}$ of links of capacity C_l ($l \in \mathcal{L}$). The network is shared by a set $\mathcal{S} = \{1, \cdots, S\}$ of sources. $\overrightarrow{R(t)} = [R_1(t), \cdots, R_S(t)]$ is the vector of rates of all sources with $R_i(t)$ denoting source i's sending rate. $\mathcal{S}_l$ is the set of sources using link l.

In each sampling period, which is the time interval between two adjacent packet sampling events, the switch monitors the queue length, the number of arriving and departing packets to calculate the congestion feedback value using Eq. (1) as in QCN. The switch also monitors the number of total bytes received for each flow $B_i(t)$ ($i \in \mathcal{S}_l$), which shares one bottleneck link l within one sampling interval at time t. The switch identifies congestion culprits by using a two-step approach, which removes the influence of low rate flows to enable precise identification of congestion culprits.

Step 1: Identify high rate flows. The fair share for each flow $i \in \mathcal{S}_l$ on link l at the sampling time t can be estimated as:

$$M_i(t) \quad = \quad \frac{W_i}{\sum_{i\in\mathcal{S}_l} W_i} \sum_{i\in\mathcal{S}_l} B_i(t) \quad (7)$$

where W_i is the weight coefficient for flow i, which can be determined by traffic class, source address, destination address, etc. Thus, the traffic flows can be classified into two categories by comparing $B_i(t)$ with $M_i(t)$ ($i \in \mathcal{S}_l$). A traffic flow i from which the total number of bytes received at the switch $B_i(t)$ is smaller than its estimated fair share $M_i(t)$ ($B_i(t) < M_i(t)$) will be assigned to the low rate source set $\mathcal{S}_l^L = \{\forall i \in \mathcal{S}_l | B_i(t) < M_i(t)\}$. Otherwise, it will be assigned to the high rate source set $\mathcal{S}_l^H = \{\forall i \in \mathcal{S}_l | B_i(t) \geq M_i(t)\}$.

Step 2: Identify congestion culprits in the high rate flow set $\mathcal{S}_l^H$. The fair share can be fine-grained among the high rate source set $\mathcal{S}_l^H$ as:

$$M_i^F(t) \quad = \quad \frac{W_i}{\sum_{i\in\mathcal{S}_l^H} W_i} \sum_{i\in\mathcal{S}_l^H} B_i(t) \quad (8)$$

Similar to the identification of high rate flows, the congestion culprits can be identified by comparing $B_i(t)$ with $M_i^F(t)$ ($i \in \mathcal{S}_l^H$). The source in the high rate source set $\mathcal{S}_l^H$ with the total number of bytes received at the switch equal to or larger than its fine-grained fair share $M_i^F(t)$ is considered as an overrated flow, which is deemed as a congestion culprit. All of overrated flows form an overrated flow set $\mathcal{S}_l^R = \{i \in \mathcal{S}_l^H | B_i(t) \geq M_i^F(t)\}$.

B. Per Flow Congestion Parameter Calculation

If the calculated congestion feedback value $F_b(t)$ using Eq. (1) is negative, the congestion notification message will be sent back to all identified overrated flows through multi-casting. For each overrated flow $i \in \mathcal{S}_l^R$, the quantized congestion feedback value $\Psi_{F_b}(i,t)$ in the congestion notification message to the source of the overrated flow $i \in \mathcal{S}_l^R$ is calculated as follows:

$$\Psi_{F_b}(i,t) \quad = \quad \frac{B_i(t)/W_i}{\sum_{k\in\mathcal{S}_l^R} B_k(t)/W_k} \times \Psi(F_b(t)) \quad (9)$$

From the above equation, we can see that the quantized congestion feedback value $\Psi_{F_b}(i,t)$ in each congestion notification message is proportional to $\Psi(F_b(t))$ and the total number of bytes received at the switch normalized by its weight coefficient $B_i(t)/W_i$, while the sum of the congestion feedback values for all congestion culprits is equal to $\Psi(F_b(t))$. Moreover, $\Psi_{F_b}(i,t)$ is also quantized to 6 bits because $F_b(t)$ is quantized to 6 bits.

Fig. 2 describes the FQCN system model with three source flows. In this example, sources 1 and 3 are identified as congestion culprits, and the congestion notification message with the quantized congestion feedback values $\Psi_{F_b}(1)$ and $\Psi_{F_b}(3)$, calculated according to Eq. (9), are fed back to sources 1 and 3, respectively.

FQCN differs from QCN as well as AF-QCN in computing the congestion feedback value. QCN does not distinguish flow-dependent congestion information and feedbacks the same

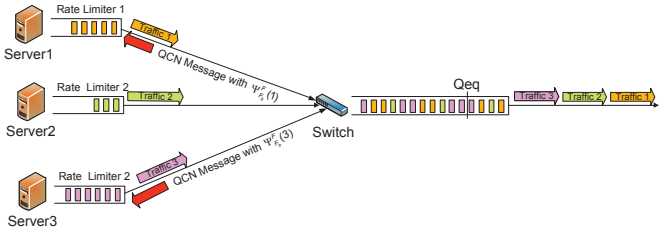


Fig. 2: FQCN overview.

congestion status to the source of the randomly sampled packet, while the congestion feedback value in the congestion notification message in both AF-QCN and FQCN is flow-dependent. FQCN distinguishes from QCN as well as AF-QCN by sending QCN congestion notification message to the sources of all congestion culprits rather than to the source of the randomly sampled packet in QCN and AF-QCN. Thus, the switch congestion is resolved much faster with FQCN than that with QCN or AF-QCN. Moreover, the signaling overhead of FQCN is lighter than that of QCN and AF-QCN because congestion messages could also be received by low rate sources, which are not the real congestion culprits. That is, throttling low rate sources are not as effective as throttling high rate sources in mitigating congestion. AF-QCN increases the congestion control algorithm complexity with arrival rate estimation, fair share calculation, and dropping probability calculation for each flow. AF-QCN can also identify congestion culprits, but it still feeds congestion information only back to the source of the randomly sampled packet.

V. Analysis of FQCN

The stability and fairness of FQCN can be analyzed via the Lyapunov method [14], which is an important tool to determine the stability of the equilibrium solution of a system of ordinary differential equations (ODEs). In this section, we analyze a simplified FQCN, in which RIT is disabled.

As introduced in Section IV, the FQCN control mechanism can be separated into two parts: CP control and RL control. At the switch, by assuming that the queue length is differentiable, the switch dynamics are given by:

$$\frac{dQ_{len}(t)}{dt} = \sum_{i\in\mathcal{S}_l} R_i(t) - C_l \tag{10}$$

Based on Eq. (1), the congestion feedback value $F_b(t)$ generated by link l can be calculated by:

$$F_b(t) = -(Q_{len}(t) - Q_{eq}) - \frac{w}{C_l * p(t)}\Big(\sum_{i\in\mathcal{S}_l} R_i(t) - C_l\Big) \tag{11}$$

where $p(t)$ is the time-varying sampling probability at the switch. This equation is the continuous version of calculating the congestion feedback value of Eq. (1). If $F_b(t)$ is negative, each congestion notification message is sent back to individual sources of the overrated flows with congestion parameter calculated by Eq. (9).

RP adjusts the sending rate by performing additive increase and multiplicative decrease (AIMD). At each rate increase event, the sending rate is increased by $(R_i^T(t) - R_i(t))/2$, and the rate increase interval is $8B_L/R_i(t)$ with BC, where $R_i^T(t)$ and $R_i(t)$ are the target and current sending rate of source i, respectively. Therefore, the expected rate increase per unit time is $\frac{(R_i^T(t)-R_i(t))R_i(t)}{16B_L}$.

Denote $p_l(f_l(t))$ as the probability that a congestion message will be sent to each overrated source $i \in S_l^R$ at each packet sampling event when the load on link l is $f_l(t)$. This probability is a monotonically increasing and differentiable function of the load [15], $f_l(t) = \sum_{j\in\mathcal{S}_l} a_{l,j}R_j(t)$, where $a_{l,j} = \{0,1\}$ indicates whether the traffic from source j flows on link l ($a_{l,j} = 1$) or not ($a_{l,j} = 0$). Note that the splitting of traffic over multiple parallel paths is not considered because the split traffic flows between the same source and destination pair can be treated as multiple flows independently. When a negative congestion message is received by RP at source i, it will reduce the sending rate by $G_d\Psi_{F_b}(i,t)R_i(t)$. The expected interval between successive congestion messages received at an overrated source is equal to $K/(C_l\sum_{l\in\mathcal{L}} p_l(f_l(t))$, where K is the packet size in bits. Therefore, the expected rate decrease per unit time is $G_d\Psi_{F_b}(i)C_l\sum_{l\in\mathcal{L}} p_l(f_l(t)R_i(t)/K$, where $\Psi_{F_b}(i)$ is the expected congestion feedback value received at source i.

Hence, an ODE describing the expected rate change at source i can be expressed as:

$$\begin{aligned}\frac{dR_i(t)}{dt} =& \frac{(R_i^T(t) - R_i(t))R_i(t)}{16B_L} \times \Big(1 - \sum_{l\in\mathcal{L}} p_l(f_l(t))\Big) \\ & - \frac{G_d\Psi_{F_b}(i)C_lR_i(t)}{K} \times \sum_{l\in\mathcal{L}} p_l(f_l(t))\end{aligned} \tag{12}$$

In order to study the attractors of the ODE Eq. (12), we identify a Lyapunov function for this ODE [14–16] as follows:

$$\sum_{l\in\mathcal{L}} p_l(f_l(t)) = \frac{\partial}{\partial R_i(t)}\sum_{l\in\mathcal{L}} P_l(f_l(t)) = \frac{\partial P\big(\overrightarrow{R(t)}\big)}{\partial R_i(t)} \tag{13}$$

where $P_l(t)$ is a primitive of $p_l(\overrightarrow{R_l(t)})$ defined by

$$P_l\big(\overrightarrow{R_l(t)}\big) = \int_0^{f_l(t)} p_l(u)\mathrm{d}u$$

and

$$P\left(\overrightarrow{R(t)}\right) = \sum_{l\in\mathcal{L}} P_l\left(f_l(t)\right)$$

Here, $P_l(f_l(t))$ is the cumulative probability of generating a negative QCN congestion notification message from link l with the link load $f_l(t)$, and $P(\overrightarrow{R(t)})$ is the cumulative probability of generating a congestion notification message from all links with the vector of rates $\overrightarrow{R(t)}$.

We can then rewrite the ODE Eq. (12) as:

$$\begin{aligned}\frac{dR_i(t)}{dt} =& \frac{R_i(t)[\alpha_i - KR_i(t)]}{16KB_L} \\ & \times \left\{\frac{K(R_i^T(t) - R_i(t))}{\alpha_i - KR_i(t)} - \frac{\partial P(\overrightarrow{R(t)})}{\partial R_i(t)}\right\}\end{aligned} \tag{14}$$

where $\alpha_i = KR_i^T + 16B_LC_lG_d\Psi_{F_b}(i)$.
The Lyapunov function for the ODE Eq. (14) is identified as:

$$J_A\big(\vec{R}\big) = \sum_{i\in\mathcal{S}} \Phi\big(R_i\big) - P\big(\vec{R}\big) \tag{15}$$

with

$$\Phi(R_i) = \int_0^{R_i} \frac{K(R_i^T - v_i)}{\alpha_i - KR_i(t)} \, \mathrm{d}v_i = R_i + \frac{16B_L C_l G_d \Psi_{F_b}(i)}{K} \log\left(1 - \frac{K}{\alpha_i} R_i\right) \quad (16)$$

The Lyapunov function Eq. (15) for the ODE Eq. (14) can be transformed to:

$$J_A(\vec{R}) = \sum_{i \in \mathcal{S}} R_i + \sum_{i \in \mathcal{S}} \frac{16B_L C_l G_d \Psi_{F_b}(i)}{K} \log\left(1 - \frac{K}{\alpha_i} R_i\right) - P(\vec{R}) \quad (17)$$

Similarly, following the same procedures as above by replacing the expected interval between successive congestion messages received at source i with $K/(R_i(t)\sum_{l \in \mathcal{L}} p_l(f_l(t)))$, a Lyapunov function $J_B(\vec{R})$ can also be identified for a QCN system as:

$$J_B(\vec{R}) = \frac{K}{K-\eta} \sum_{i \in \mathcal{S}} \left[R_i + \frac{\eta R_i^T}{K-\eta} \log\left(1 - \frac{K-\eta}{KR_i^T} R_i\right)\right] - P(\vec{R}) \quad (18)$$

with $\eta = 16B_L G_d \Psi_{F_b}$, where Ψ_{F_b} is the expected congestion feedback value.

A. Stability analysis

Proposition 1: The strictly concave function $J_A(\vec{R})$ is a Lyapunov function for the differential equation of the FQCN system Eq. (14), which is stable, and the unique value $\overrightarrow{R}$ that maximizes $J_A(\vec{R})$ is an equilibrium point of the system, to which all trajectories converge and maximize the link utilization.

Proof: From Eq. (17), it is easy to see that $J_A(\vec{R})$ is strictly concave over the bounded region ($\vec{R} \succeq 0$) with an interior maximum, and therefore the value of $\overrightarrow{R}$ that maximizes $J_A(\vec{R})$ is unique. The equilibrium rates $\overrightarrow{R}$ that maximizes $J_A(\vec{R})$ can be identified by setting the following derivatives to zero:

$$\frac{\partial J_A(\overrightarrow{R(t)})}{\partial R_i(t)} = \frac{K(R_i^T(t) - R_i(t))}{\alpha_i - KR_i(t)} - \frac{\partial P(\overrightarrow{R(t)})}{\partial R_i(t)} \quad (19)$$

Further, along any solution of $\overrightarrow{R(t)}$, we have:

$$\frac{d}{dt} J_A(\overrightarrow{R(t)}) = \sum_{i \in \mathcal{S}} \frac{\partial J_A(\overrightarrow{R(t)})}{\partial R_i(t)} \frac{dR_i(t)}{dt} = \sum_{i \in \mathcal{S}} \frac{R_i(t)[\alpha_i - KR_i(t)]}{16KB_L} \left(\frac{\partial J_A(\overrightarrow{R(t)})}{\partial R_i(t)}\right)^2 \quad (20)$$

Thus, $J_A(\overrightarrow{R(t)})$ is strictly increasing with t, unless $\overrightarrow{R(t)} = \overrightarrow{R}$, the unique $\overrightarrow{R}$ that maximizes $J_A(\vec{R})$. The function $J_A(\vec{R})$ is thus a Lyapunov function for the differential equations of the FQCN system Eq. (14). The maximization of $J_A(\vec{R})$ is constrained by the link capacity, and hence the equilibrium rates $\overrightarrow{R}$ maximize the link utilization. Since the equilibrium rates are stable, the queue is also stable. Similar results can be observed with the Lyapunov function $J_B(\vec{R})$ for the QCN system. Hence, both FQCN and QCN are stable, and the rate of each source converges to an equilibrium point that maximizes the link utilization.

B. Fairness analysis of FQCN

Proposition 2: With multiple sources sharing a link, FQCN is closer to max-min fairness than to proportional fairness.

Proof: As shown in [15], the expectation of the congestion feedback $p_l(f_l)$ at link l is close to a Dirac function in the limiting case as:

$$\delta_{C_l}(f_l) = \begin{cases} 0 & \text{if } (f_l < C_l) \\ +\infty & \text{if } (f_l \geq C_l) \end{cases}$$

Thus, following the method in [15], the rates with FQCN are distributed so as to maximize:

$$F_A(\vec{R}) = \sum_{i \in \mathcal{S}} \frac{16B_L C_l G_d \Psi_{F_b}(i)}{K} \log\left(1 - \frac{K}{\alpha_i} R_i\right) \quad (21)$$

subject to the constraints:

$$\sum_{l \in \mathcal{L}} a_{l,i} R_i \leq C_l, \forall l \in \mathcal{L}$$

Similarly, in a QCN system, the rates are distributed to maximize:

$$F_B(\vec{R}) = \frac{K\eta}{(K-\eta)^2} \sum_{i \in \mathcal{S}} R_i^T \log\left(1 - \frac{K-\eta}{KR_i^T} R_i\right) \quad (22)$$

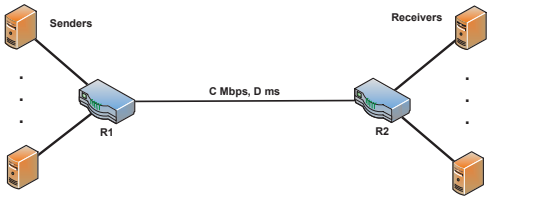
At an equilibrium point, R_i is very close to R_i^T. Hence, we can observe that in a QCN system, the weight given to R_i is close to $K\eta/(K-\eta)^2 R_i^T \log(\eta/K)$ for large R_i, while the weight given to R_i in a FQCN system tends to $-16B_L G_d \Psi_{F_b}(i) C_l/K \log(1 + 16B_L G_d \Psi_{F_b}(i)/K)$ as R_i tends to C_l. Thus, FQCN is closer to max-min fairness than to proportional fairness. This conforms to the intuition of FQCN: low rate sources are less likely to receive QCN congestion notification messages, while overrated flows will decrease the sending rate according to the quantized congestion feedback value in the congestion notification messages.

VI. Performance Evaluation

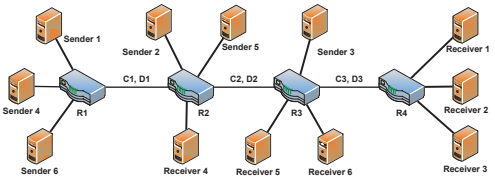
In this section, we evaluate FQCN with different traffic dynamics by using NS2 [17] under three network topologies, namely, the dumb-bell topology, parking-lot topology and a simple and representative TCP Incast network topology. We compare the performance of FQCN with that of QCN and AF-QCN in terms of fairness and convergence, and the effects of these congestion notification algorithms on TCP Incast. The default QCN configuration is used in our evaluations: w=2, G_d=1/128, T=15 ms, C_T = 5, B_L = 150 KB, R_{AI}=5 Mbps and R_{HAI}=50 Mbps when the link capacity of the switch is 10 Gbps, while R_{AI}=0.5 Mbps and R_{HAI}=5 Mbps when the link capacity of the switch is 1 Gbps.

A. Simulation Topologies

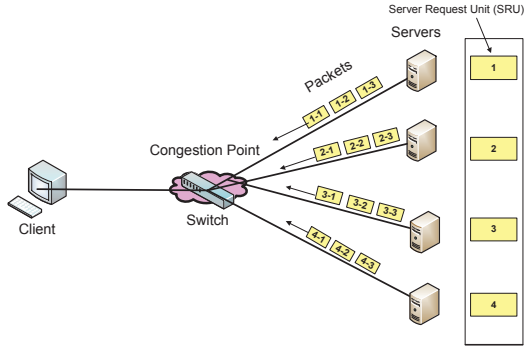
The dumb-bell and parking-lot topologies are shown in Fig. 3(a) and 3 (b), respectively. In these two topologies, all the links have 10 Gbps link capacity and 50 μs round-trip time (RTT) delay unless otherwise stated. 150 kilobytes (KB) of switch buffers are used and the equilibrium queue length Q_{eq} is set to 33 KB. Furthermore, we simulate the application performance with synchronized reads over TCP in NS2 to model a typical striped file system data transfer operation to



(a) Dumbbell



(b) Parkinglot



(c) A simple and representative TCP Incast network setting with one client requesting data from multiple servers through synchronized reads.

Fig. 3: Simulation topologies.

test the effects of QCN, AF-QCN and FQCN on the TCP Incast problem.

TCP Incast [18, 19], also called TCP throughput collapse, has been observed in many data center applications, e.g., in cluster storage [20], when storage servers concurrently respond to requests for a data block, in web search, when many workers respond near simultaneously to search queries, and in batch processing jobs like MapReduce [21], in which intermediate key-value pairs from many Mappers are transferred to appropriate Reducers during the "shuffle" stage. TCP Incast is attributed to having multiple senders overwhelming a switch buffer, thus resulting in TCP timeout due to packet drops at the congestion switch as analyzed in [19, 22, 23]. A simple and basic representative network topology in which TCP throughput collapse can occur is shown in Fig. 3(c). Data is stripped over a number of servers, and stored as a server request unit (SRU) on each server. In order to access one particular data block, a client needs to perform synchronized readings: sending request packets to all of the storage servers containing a fragment of data block for this particular block. The client will not generate data block requests until it has received all the data for the current block. Upon receiving the requests, the servers transmit the data to the receiver through one Ethernet switch almost concurrently. Small Ethernet buffers may be exhausted by these concurrent flood of traffic, thus resulting in packet loss and TCP timeouts. Therefore, TCP Incast may be observed during synchronized readings for data blocks across an increasing number of servers. In this testing scenario, all the links have 1 Gbps link capacity and 100 μs RTT delay and the switch buffer size is 64KB. In order to validate the simulation results, all of these experimental parameters are configured as the same as one of those experiments conducted in [19], which explored the TCP Incast problem with various configuration parameters, such as buffer size, SRU size, and TCP variants.

B. Simulation Results

1) Backlogged Static Traffic: We conduct several experiments with static backlogged flows in the dumbbell topology and parking-lot topology to validate the fairness improvement and queue length stability of FQCN. The static backlogged traffic sources in our evaluations are simulated with constant bit rate (CBR) traffic flows carried by User Datagram Protocol (UDP), unless otherwise stated.

In the first experiment, four static flows are initiated simultaneously to traverse through the single bottleneck in the dumb-bell topology. The total simulation time is 6 seconds, and the switch service rate is reduced from 10 Gbps to 1 Gbps and changed back to 10 Gbps at 2 and 4 second of the simulation time, respectively. The flow rates of individual flows and the switch queue length for QCN, AF-QCN and FQCN are shown in Fig. 4. Note that the four sources achieve very different rates by QCN, showing the unfairness of QCN. In this experiment, AF-QCN and FQCN can allocate their equal fair share rates successfully while maintaining the queue length stability. The sending rates with FQCN stabilize to their fair share rates faster and smoother than those of AF-QCN. The standard deviation of sending rates for AF-QCN are 3-4 times larger than that of FQCN.

In the parking-lot topology, six static source flows are initiated one after another at an interval of 1 second. The sending rates of each individual flow with QCN, AF-QCN and FQCN are shown in Fig. 5. All of these experiments further confirm the successful allocation of fair share rates with AF-QCN and F-QCN while the queue length stability is also

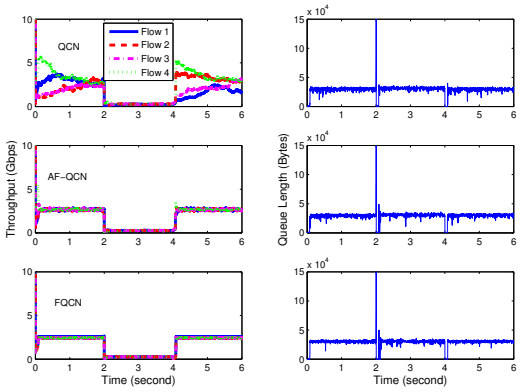


Fig. 4: The average throughput of 4 simultaneous static sources (left) and switch queue length (right) in the dumbbell topology.

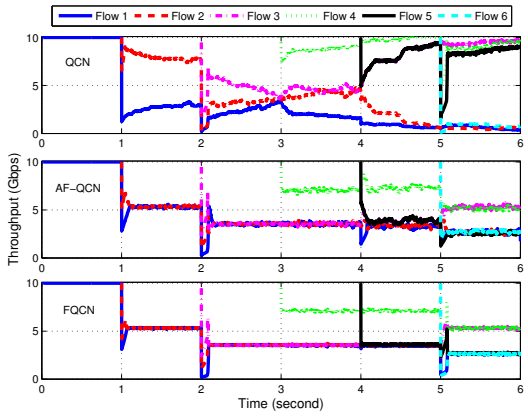
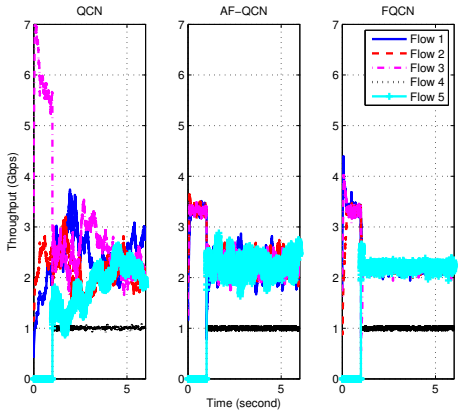
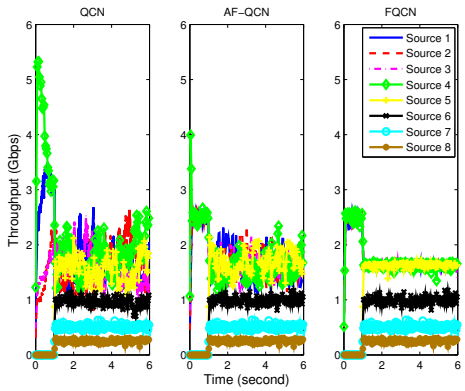


Fig. 5: The average throughput in the parking-lot topology.

maintained which is not shown here owing to the space limit. Again, the flow sending rates with FQCN stabilize to their fair share rates faster and smoother than those with AF-QCN. All of these experiments validate our propositions in Section V that FQCN is stable and the rates converge to maximize the link utilization.



(a) Mix static and burst traffic



(b) Mix static and poisson traffic

Fig. 6: The average throughputs for individual flows of mix traffic in the dumb-bell topology.

2) Mix of Static and Dynamic Traffic: Fig. 6 (a) shows the average throughput of mix traffic of three static flows and two ON-OFF burst traffic flows in the dumb-bell topology. The average offered traffic loads for these two ON-OFF burst traffic flows are 1 Gbps and 5 Gbps, respectively. All the static flows are initiated at 0 second and the burst traffic flow is initiated at 1.0 second. The "ON" period of burst traffic flow is determined by completing 10 KB data transfer, and the "OFF" period is chosen to meet the average burst traffic load. The average throughput of the burst flow with lower load is equal to its 1 Gbps burst traffic load since its offered load is lower than its fair share. If the offered burst traffic load is larger than its fair share, its throughput will be limited to its fair share. As seen in the experiment with 5 Gbps burst traffic load, the average throughput for this burst flow is limited to its fair share 2.25 Gbps. The simulation results in this experiment also validate again that the left over capacity is shared by other three static flows unfairly with QCN and almost equally fairly with AF-QCN and FQCN, and the average throughput is more stable to reach its fair share with FQCN than that with AF-QCN.

Fig. 6 (b) shows the average throughput of mix traffic of backlogged static flows and dynamic flows, whose arrival rates are Poisson distributed, in the dumb-bell topology with eight source-destination pairs under different congestion notification algorithms. The first four sources are backlogged static flows, and the other four dynamic sources are built up with [8 8 4 2] connections at the flow arrival rates of [250 125 125 125] Mbps and the flow size with Pareto distribution with a mean of 10 KB and a shape parameter of 1.1, respectively. Thus, these four dynamic sources offer [2.0 1.0 0.5 0.25] Gbps traffic load in total, respectively. Simulation results show that the average throughput for each individual dynamic flow is around [1.6 1.0 0.5 0.25], respectively. The average throughput for dynamic source with 2.0 Gbps total traffic load is overloaded and is limited to its fair share 1.65 Gbps. The other dynamic sources are light-loaded sources whose traffic loads are smaller than their equal fair shares, and therefore the average throughput for these light-loaded dynamic sources are equal to their offered load. Note that FQCN successfully allocates the residual capacity, left over from the light-loaded dynamic sources, equally fair among four backlogged static sources and one overloaded dynamic source, and each individual source quickly converges to and stays stably at its fair share.

3) Effect of Congestion Notification Algorithms to TCP Incast: Fig. 7 depicts the TCP throughput collapse for a synchronized read application performed on the network shown in Fig. 3 (c). As a comparison, TCP goodput without any congestion control is also shown in the same figure. In this simulation, a SRU size of 256 KB is chosen to model a production storage system, the default Minimum Timeout Retransmission (RTO) is 200 ms, TCP NewReno implementation with the default packet size of 1000 bytes is used in the simulation, and the equilibrium queue size is set to 14 KB. From the simulation results, we can see that without any congestion control, TCP goodput drops very quickly with the increase of the number of servers. Note that QCN and AF-QCN perform poorly in the TCP Incast setup (Fig. 3 (c)). QCN can delay the occurrence of TCP Incast, but the TCP goodput also decreases with the

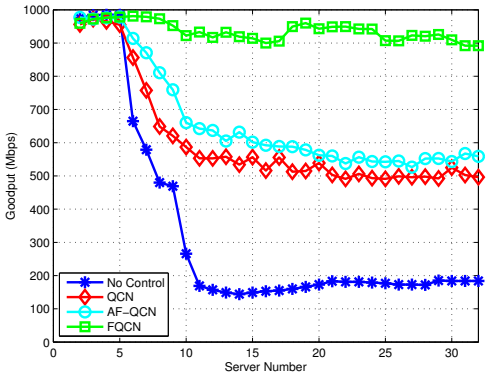


Fig. 7: TCP throughput for a synchronized reading application.

increase of the number of servers, reducing to around 500 Mbps with 16 servers. AF-QCN achieves a TCP goodput of 50 Mbps more than QCN. On the other hand, FQCN is able to mitigate TCP Incast superbly without much goodput degradation, maintaining a high goodput of around 900 Mbps even with a large number of servers, as shown in Fig. 7.

The poor performance of TCP throughput with QCN and AF-QCN is attributed to the rate unfairness of different flows. We examine the rate fluctuation of different flows within one synchronous reading request with QCN and confirm that some traffic flows have higher allocated sending rates than other flows. After these high rate flows finish the data transfer, other traffic flows attempt to increase their sending rates to recover the available bandwidth previously utilized by high rate flows. It takes several milliseconds to recover this part of available bandwidth by other traffic flows; during this period, the bandwidth is not fully utilized. Hence, the unfair rate allocation with QCN results in poor TCP throughput in this TCP Incast setting.

Similar unfair rate allocation to different flows in this TCP Incast setup with AF-QCN can also be observed, but with smaller sending rate variations among different traffic flows and smaller time interval between the first and last server in completing the data transfer than that with QCN in one synchronous reading request. We further examine what cause the rate unfairness of different flows in TCP Incast setup with AF-QCN. Similar to QCN, AF-QCN only feeds the congestion notification message back to the source of the sampled packet. It takes some time to let all sources receive the congestion information, and so the later the source gets the congestion notification message, the more likely the source experiences packet drops. As a transmission control protocol, TCP exercises a congestion control mechanism itself. When packet drops occur, TCP slows down its sending rate by adjusting the congestion window size and slow start threshold. AF-QCN only regulates Layer 2 sending rate and attempts to allocate bandwidth fairly among all sources by taking into account of the flow-dependent congestion information in the congestion notification message. However, TCP does not take any flow-dependent congestion information to adjust its congestion window size and slow start threshold. This causes the rate unfairness of different flows in TCP Incast setup with AF-QCN eventually.

We also examine the rates of all flows within one barrier synchronous reading request with the FQCN congestion control algorithm, which facilitates fair sharing of the link capacity among all the source flows since all the overrated flows receive the congestion notification messages at almost the same time.

4) Different Traffic Weights: FQCN can also allocate weighted fair share rates as well as AF-QCN. The flows rates in the dumb-bell topology with FQCN and weight coefficient settings [4, 3, 2, 1] for 4 static simultaneous flows are shown in Fig. 8. The larger the weight coefficient, the larger share rate will be allocated to the flow. In this simulation, the maximum sending rate for the first flow whose weight coefficient is set to 4 will be cut down from 10 Gbps to 1 Gbps at the third second of the simulation time. From the simulation results, we can see that before the maximum sending rate changes, the flow rates are allocated fairly consistent to their weight coefficients. After the third second, the sending rate of the first flow is limited by its maximum sending rate of 1 Gbps, and the leftover link capacity is shared by other traffic flows fairly consistent to their weight coefficients.

VII. Conclusion

We have proposed an enhanced QCN congestion notification algorithm, called FQCN, to improve fairness allocation of bottleneck link capacity among all sharing sources. We have evaluated the performance of FQCN with different traffic dynamics by simulations under three network topologies, namely, the dumb-bell topology, parking-lot topology and a simple and representative TCP Incast network topology, and compared the performance of FQCN with that of QCN and AF-QCN. The simulation results have shown that FQCN can successfully allocate the fair share rate to each traffic source while maintaining the queue length stability. As compared to AF-QCN, the flow sending rates with FQCN are more stable with respect to the fair share rates. FQCN can also provide weighted fair share allocation of the bottleneck link capacity. Moreover, FQCN significantly enhances TCP throughput performance in the TCP Incast setting, and achieves better

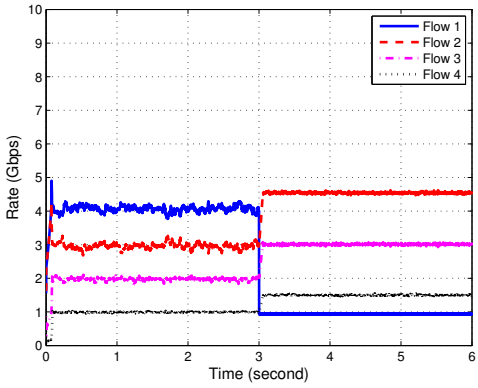


Fig. 8: Flows rates in the dumb-bell topology with FQCN and weight coefficient settings [4, 3, 2, 1] for 4 static flows.

TCP throughput than QCN and AF-QCN. FQCN performance under other traffic conditions will be further studied in the future.

Finally, millions of flows co-existing in a DCN may raise the scalability problem for FQCN due to the need of estimating per-flow information. A distributed flow monitoring system can be helpful to solve this problem, which is left for our future work.

REFERENCES

[1] "Data Center Bridging Task Group," http://www.ieee802.org/1/pages/dcbridges.html.

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

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

[4] J. Jiang, R. Jain, and C. So-In, "An Explicit Rate Control Framework for Lossless Etherent Operation," in *International Conference on Communication 2008*, Beijing, China, May 19-23, 2008, pp. 5914 – 5918.

[5] C. So-In, R. Jain, and J. Jiang, "Enhanced Forward Explicit Congestion Notification (E-FECN) Scheme for Datacenter Ethernet Networks," in *International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, Edinburgh, UK, Jun. 2008, pp. 542 – 546.

[6] M. Alizadeh, B. Atikoglu, A. Kabbani, A. Lakshmikantha, R. Pan, B. Prabhakar, and M. Seaman, "Data Center Transport Mechanisms: Congestion Control Theory and IEEE Standardization," in *the 46th Annual Allerton Conference*, Illinois, USA, Sep. 2008, pp. 1270–1277.

[7] "QCN pseudo-code version 2.0," http://www.ieee802.org/1/files/public /docs2008/au-rong-qcn-serial-hai-pseudo-code%20rev2.0.pdf.

[8] M. Alizadeh, A. Kabbani, B. Atikoglu, and B. Prabhakar, "Stability Analysis of QCN: The Averaging Principle," in *Proc. of SIGMETRICS'11*, 2011.

[9] J. Jiang and R. Jain, "Analysis of Backward Congestion Notification (BCN) for Ethernet In Datacenter Applications," in *Proc. of IEEE INFOCOM'07*, Alaska, USA, May 6-12, 2007, pp. 2456–2460.

[10] P. Devkota and A. Reddy, "Performance of Quantized Congestion Notification in TCP Incast Scenarios of Data Centers," in *Proc. of 2010 IEEE International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems*, Aug. 2010, pp. 235 –243.

[11] Y. Zhang and N. Ansari, "On Mitigating TCP Incast in Data Center Networks," in *Proc. of IEEE INFOCOM'11*, Shanghai, China, April 2011, pp. 51 –55.

[12] A. Kabbani, M. Alizadeh, M. Yasuda, R. Pan, and B. Prabhakar, "AF-QCN: Approximate Fairness with Quantized Congestion Notification for Multi-tenanted Data Centers," in *Proc. of IEEE the 18th Annual Symposium on High Performance Interconnects*, Aug. 2010, pp. 58 –65.

[13] Y. Zhang and N. Ansari, "On Architecture Design, Congestion Notification, TCP Incast and Power Consumption in Data Centers," *IEEE Communications Surveys Tutorials*, vol. 15, no. 1, pp. 39–64, 2013.

[14] Jean-Yves Boudec, *Rate adaptation, Congestion Control and Fairness: A Tutorial*. Ecole Polytechnique Fédérale de Lausanne, December 2000.

[15] F.P. Kelly, A.K. Maulloo, and D.K.H. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *The Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252, March 1998.

[16] S. Golestani and S. Bhattacharyya, "A class of end-to-end congestion control algorithms for the internet," in *Proc. of the Sixth International Conference on Network Protocols, 1998*, Austin, Texas, USA, Oct. 13-16, 1998, pp. 137 –150.

[17] ns-2 Network Simulator, http://www.isi.edu/nsnam/ns/.

[18] D. Nagle, D. Serenyi, and A. Matthews, "The Panasas ActiveScale Storage Cluster: Delivering Scalable High Bandwidth Storage," in *ACM/IEEE conference on Supercomputing*, Washington, DC, 2004.

[19] A. Phanishayee, E. Krevat, V. Vasudevan, D. G. Andersen, G. R. Ganger, G. A. Gibson, and S. Seshan, "Measurement and analysis of TCP throughput collapse in cluster-based storage systems," in *the 6th USENIX Conference on File and Storage Tech.*, San Jose, CA, 2008, pp. 1–14.

[20] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 29–43, 2003.

[21] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[22] Y. Chen, R. Griffith, J. Liu, R. H. Katz, and A. D. Joseph, "Understanding TCP Incast Throughput Collapse in Datacenter Networks," in *Proc. of the 1st ACM workshop on Research on enterprise networking*, Barcelona, Spain, August 21, 2009, pp. 73–82.

[23] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, and B. Mueller, "Safe and effective fine-grained TCP retransmissions for datacenter communication," in *ACM SIGCOMM'09*, Barcelona, Spain, August 21, 2009, pp. 303–314.

Yan Zhang (S'10) received the B.E. and M.E. degrees in Electrical Engineering from Shandong University, P.R. China, in 2001 and 2004, respectively. She is currently pursuing her Ph.D degree in Computer Engineering in the Department of Electrical and Computer Engineering at the New Jersey Institute of Technology (NJIT), Newark, New Jersey. Her research interests include congestion control and energy optimization in data center networks, content delivery acceleration over wide area networks, and energy efficient networking.

Nirwan Ansari (S'78-M'83-SM'94-F'09) is Professor of Electrical and Computer Engineering at NJIT. He has contributed over 450 technical papers, over one third published in widely cited refereed journals/magazines. He has been granted over twenty US patents. He has guest-edited a number of special issues, covering various emerging topics in communications and networking. His current research focuses on various aspects of broadband networks and multimedia communications. He has assumed various leadership roles in IEEE Communications Society, and some of his recognitions include Thomas Edison Patent Award (2010), NJIHoF Inventor of the Year Award (2012), and a couple of best paper awards.