Computer Networks 130 (2018) 51–64

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Analysing and improving convergence of quantized congestion notification in Data Center Ethernet

Ran Shu[a], Fengyuan Ren[a,*], Jiao Zhang[b], Tong Zhang[a], Chuang Lin[a]

[a] Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, China
[b] School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, 100876, China

ARTICLE INFO

Article history:
Received 12 November 2016
Revised 17 October 2017
Accepted 14 November 2017
Available online 21 November 2017

Keywords:
Data Center Ethernet
Quantized Congestion Notification
Convergence
Modeling

ABSTRACT

Quantized Congestion Notification (QCN) has been approved as the standard congestion management mechanism for the Data Center Ethernet (DCE). However, lots of work pointed out that QCN suffers from the problem of unfairness among different flows. In this paper, we found that QCN could achieve fairness, merely the convergence time to fairness is quite long. Thus, we build a convergence time model to investigate the reasons of the slow convergence process of QCN. We validate the precision of our model by comparing with experimental data on the NetFPGA platform. The results show that the proposed model accurately well characterizes the convergence time to fairness of QCN. Based on the model, the impact of QCN parameters, network parameters, and QCN variants on the convergence time is analysed in detail. Results indicate that the convergence time of QCN can be decreased if sources have the same rate increase probability or the rate increase step becomes larger at steady state. Enlightened by the analysis, we proposed a mechanism called QCN-T, which replaces the original Byte Counter and Timer at sources with a single modified Timer to reduce the convergence time. Finally, evaluations show great improvements of QCN-T in both convergence and stability.

1. Introduction

Recently, using a unified infrastructure to replace LAN, SAN and HPC networks in data centers has attracted much attention [1,2]. Data Center Ethernet (DCE), also called Converged Enhanced Ethernet (CEE) or Data Center Bridging (DCB), is considered as the predominant choice for the unified infrastructure due to Ethernet's features of easy management, low cost, and so on. SANs and HPC networks require lossless transfer as well as low end-to-end delays. To satisfy these requirements, DCE needs to enhance the performance of traditional Ethernet. Congestion control in DCE, designed by the IEEE 802.1Qau work group, is one critical enhancement [2]. It aims to provide end-to-end congestion management for traffic without congestion control mechanisms above the link layer, such as Fibre Channel over Ethernet (FCoE), UDP. Also, it is expected to benefit up-layer protocols at coexistence, such as TCP, whose congestion control mechanisms do not perform very well in data centers.

The QCN protocol has been ratified to be the standard congestion management scheme for DCE in March 2010 [2]. Many new switches have been designed to support QCN functions, such as Cisco Nexus 7000 [3] and FocalPoint FM6000 [4].

However, some recent work points out that flows could not obtain their fair share of bandwidth under QCN [5–7]. The unfairness of QCN will negatively impact the performance of services running over DCE. For example, the MapReduce programming model is widely employed by services in today's data centers [8,9]. A large job will be partitioned into small tasks and assigned to different workers. The final completion time of the job is determined by the slowest worker. Therefore, if all workers could not get their fair share of bandwidth, the flow completion time of them will have large variance and thus the job will be lagged by the sluggish worker.

Some attempts have been made to explore the reasons for QCN unfairness. There are two main points. First, a Reaction Point (RP) decreases its rate upon receiving a negative feedback from a Congestion Point (CP). However, a CP transmits each feedback message to a *randomly* selected RP. The random destinations of feedback incurs unfairness among RPs. Second, the flows with higher rates have more opportunities to increase their rates [6]. RPs use both Byte Counter and Timer to control rate increase in QCN. If the Byte

* Corresponding author.
E-mail addresses: shuran@csnet1.cs.tsinghua.edu.cn (R. Shu), renfy@tsinghua.edu.cn (F. Ren), jiaozhang@bupt.edu.cn (J. Zhang), zhangt14@mails.tsinghua.edu.cn (T. Zhang), chlin@tsinghua.edu.cn (C. Lin).

https://doi.org/10.1016/j.comnet.2017.11.004
1389-1286/

Counter of an RP shows that the RP has transmitted 150 KB data or the Timer has passed 15 ms, the rate of the RP will increase. Generally the Byte Counter dominates the rate increase in DCE. Thus, the RPs with larger sending rates will increase their rates more quickly.

Based on the two kinds of reasons, some mechanisms are proposed to make QCN more fair. To avoid unfair feedback, AF-QCN [5] and FQCN [7] have been proposed to fairly transmit feedbacks to each RP. However, these two mechanisms require that each switch maintain the information of all the passing flows, which disobeys the design principle that QCN switch does not save flow information. To avoid that RPs with higher rates increase quickly, the Byte Counter employed in QCN is modified to an adaptive Byte Counter [6]. However, it is difficult to determine a general parameter value used in the adaptive Byte Counter.

To solve the problem of long convergence time in QCN at low cost, we need to thoroughly understand the radical reasons for the long convergence time in depth. In this paper, we first investigate the whole convergence process of QCN using experiments in a small testbed which consists of Dell Servers and NetFPGA. We find that QCN is actually fair. However, the convergence time to fairness is quite long, which possibly exceeds the investigation time in former work. Hence, many researchers stated QCN is not fair.

After investigating the experimental data, we conclude that the convergence process of QCN could be partitioned into three stages. The first two stages determine the initial rate values of the 3rd stage, and the 3rd stage will lead to the long convergence time of QCN if the initial rates are not fair. The reasons for unfairness during the first two stages are straightforward. Therefore, our model on QCN convergence time to fairness mainly characterizes the rate evolution during the 3rd stage. The proposed model indicates that if the probability of rate increase at RPs is irrespective of the current rate values of RPs or the rate increase value per time could be larger, then the convergence time of QCN can be reduced.

The proposed model is evaluated and compared against the experimental data on the NetFPGA platform. The results show that the model well characterizes QCN convergence time to fairness. Furthermore, the impact of QCN parameters, network configurations, and different QCN variants on QCN convergence time is also analysed.

Enlightened by the experimental investigation results and model analysis, we conclude that using Timer can definitely reduce QCN convergence time. Therefore, we propose QCN-T, an enhanced mechanism that replaces the Byte Counter and Timer in the standard QCN with a single modified Timer to control the rate increase. Experimental evaluations show that our proposed QCN-T could significantly decrease the convergence time to fairness compared with QCN. Besides, QCN-T reaches a fairer bandwidth share. Moreover, analysis shows QCN-T also improves stability when the number of senders is larger than 5. Further experiments reveal that QCN-T has more than twice the stability margin as QCN in the typical Data Center Ethernet environment.

The remainder of this paper is organized as follows. Section 2 introduces the background. In Section 3, we investigate the rate evolution process of QCN through experiments on the NetFPGA platform. Detailed convergence time model of QCN is described in Section 4. In Section 5, the accuracy of our model is validated by comparing the analysis results with experiment data on NetFPGA platform, and the impact of different factors on the convergence time of QCN is also discussed with the model. Section 6 proposes the enhanced mechanism — QCN-T. We evaluate the convergence and stability of QCN-T through further experiments in Section 7. Finally, the paper is concluded in Section 8.

2. Background

In this section, we briefly describe the QCN mechanism, focusing on those parts that are relevant to our analysis. The whole description can be seen in [10,11]. QCN is composed of two parts.

- *Switch or CP*. CP samples packets and generates feedback frames according to the queue length information. Feedback frames are sent to the source of packets directly.
- *Rate Limiter or RP*. RP decreases its sending rate based on feedback, and probes for available bandwidth by self-increase.

2.1. The CP algorithm

The goal of CP is to maintain the queue at a desired length Q_{eq}. CP samples incoming packets with a period whose duration is related to congestion. Normally the period is the duration of transmitting 150 KB data. Let Q denote the current queue length and Q_{old} denote the queue length of last sampling. CP calculates f_b as follows:

$$f_b = -(Q_{off} + w * Q_\delta), \tag{1}$$

where $Q_{off} = Q - Q_{eq}$, $Q_\delta = Q - Q_{old}$, w is a constant weight value. It is set to be 2 in the baseline implementation. Thus, both of the buffer excess and the rate excess are captured. Negative f_b means that there is congestion or congestion is going to happen. The value of f_b is quantized to a 6 bits value F_b, and a feedback frame containing F_b will be sent to the source of this sampled packet. If f_b is positive, no feedback frame will be sent.

2.2. The RP algorithm

RPs decrease their sending rates upon receiving negative feedback frames from CPs. Since there is no positive feedback from CPs to make RPs increase their rates, RPs have a self-increasing algorithm. Thus, rate decreases and rate increases are separated in RP. The design of rate increases at RPs originates from BIC-TCP[12]. Let Current Rate (R_c) denote the sending rate of RP and Target Rate (R_t) denote the sending rate just before the arrival of the last feedback frame. R_t is used to control rates more accurately.

Rate decreases: When a feedback frame is received, RPs update R_t and R_c as follows:

$$\begin{cases} R_t = R_c, \\ R_c = R_c(1 - G_d F_b), \end{cases} \tag{2}$$

where G_d is chosen so that $G_d F_{bmax} = 0.5$, i.e. G_d is $\frac{1}{128}$ in the baseline implementation.

Rate increases: The rate increase interval is controlled by the cooperation of Byte Counter and Timer at RPs. Each cycle of Byte Counter is 150 KB and Timer is 15 ms in 1 Gbps baseline implementation. Each cycle of Byte Counter or Timer leads to a rate increase operation. The rate increase of RPs has three phases:

Fast Recovery (FR): After a rate decrease, both Byte Counter and Timer are reset. RP tries to get the lost rate back. At the end of each cycle, R_t remains unchanged while R_c is updated as follows:

$$R_c = \frac{1}{2}(R_c + R_t). \tag{3}$$

Active Increase (AI): With either Byte Counter or Timer larger than 5, RP enters the AI phase to probe for extra bandwidth. The duration of each cycle is cut by half in FR i.e. 75 KB for Byte Counter and 7.5 ms for Timer for a more frequent probing. At the end of each cycle, R_t is added by a constant value while R_c is updated the same as in FR:

$$\begin{cases} R_t = R_t + R_{ai}, \\ R_c = \frac{1}{2}(R_c + R_t). \end{cases} \tag{4}$$