ISLAMIC UNIVERSITY OF TECHNOLOGY(IUT)
Department of Computer Science and Engineering

Mustakim Billah
Student Id: 094408
Computer Science & Engineering
Islamic University of Technology
Board Bazar, Gazipur
E-mail: mustakim.iut@gmail.com

Md. Humaun Kabir
Student Id: 094426
Computer Science & Engineering
Islamic University of Technology
Board Bazar, Gazipur
E-mail: humayunkabirnoyan@ymail.com

Congestion Avoidance and Synchronized Rate Control in Data Center Network

B.Sc Thesis
October, 2013

Supervisor: Shahriar Kaisar
Assistant Professor
Computer Science & Engineering
Islamic University of Technology
Board Bazar, Gazipur
E-mail: shahriar.kaisar@gmail.com

Acknowledgement

At first we would like to thank our respected supervisor Shahriar Kaisar, Assistant Professor,CSE Department for his guidence, supervision and earnest support through out the whole thesis work.We are grateful to our honorable Dr. Muhammad Mahbub Alam, Associate Professor, CSE Department, for his guidance and co-operation. We thank all the teacher of CSE department for their support and enthusiasm.

October, 2013

Mustakim Billah

Md. Humaun Kabir

Table of Contents

List of Figures

Abstract

TCP Incast, also known as TCP throughput collapse, is a term used to describe a link capacity under-utilization phenomenon in certain many-to-one communication patterns, typically in many datacenter applications. The main root cause of TCP Incast analyzed by prior works is attributed to packet drops at the congestion switch that result in TCP timeout. Congestion control algorithms have been developed to reduce or eliminate packet drops at the congestion switch. In this paper, the performance of Quantized Congestion Noti?cation (QCN) with respect to the TCP incast problem during data access from clustered servers in datacenters are investigated. QCN can effectively control link rates very rapidly in a datacenter environment. However, it performs poorly when TCP Incast is observed. To explain this low link utilization, we examine the rate ?uctuation of different ?ows within one synchronous reading request, and ?nd that the poor performance of TCP throughput with QCN is due to the rate unfairness of different ?ows. Therefore, an enhanced QCN congestion control algorithm, called fair Quantized Congestion Noti?cation (FQCN), is proposed to improve fairness of multiple ?ows sharing one bottleneck link. We evaluate the performance of FQCN as compared to that of QCN in terms of fairness and convergence with four simultaneous and eight staggered source ?ows. As compared to QCN, fairness is improved greatly and the queue length at the bottleneck link converges to the equilibrium queue length very fast. The effects of FQCN to TCP throughput collapse are also investigated. Simulation results show that FQCN signi?cantly enhances TCP throughput performance in a TCP Incast setup.

Chapter 1

Introduction

1.1 Data Center

Information is key to an enterprise's competitiveness. As network and communication Technologies develop at an ever increasing rate, data centers (DCs) have become the core of the information an enterprise needs to do business. A well-designed data center will improve efficiency and development of enterprises.
The DC of an enterprise is the important as it hosts key service systems, and is a center where the key data of the enterprise is managed. It controls user access, filters packets for security, Processes service applications, computes information, and stores data for backup.
A DC consists of the following components:

1. Equipment room
2. Power supply system
3. Network devices including devices on the data network, computing network, and storage network
4. Servers including operating systems and application software
5. Storage devices
6. Security system
7. Operation, administration, and maintenance (OAM) system

For enterprises, the trend is to integrate services and data in multiple DCs. This requires the enterprise network to have high level of performance and reliability.
The Huawei DC network solution provides a high performance, secure, and reliable network, which allows the DC to transmit high-quality services.

1.2 Overall Requirement for the DC Network:

A DC has a large number of servers deployed and is not only the logical center of an enterprise network but also the source of services. Therefore, a DC should provide abundant bandwidth resources, secure and reliable devices, high-quality network management, and comprehensive value-added services. To create as much value as possible based on limited bandwidth when designing and constructing the DC network, focus on the following requirements:

1.2.1 Reliability

High reliability ensures successful operations of the DC. If the user experience on enterprise services (such as e-commerce or video services) deteriorates due to DC network faults, the service expansion of an enterprise will be hindered, and users will not use the services, decreasing the profits. Reliability is an important aspect when designing an enterprise DC network. The reliability design is achieved through redundant links, key devices, and key service modules.

1.2.2 Scalability

Each layer of the DC uses devices with a high port density to prepare for the DC expansion. Devices on the Internet layer, intranet layer, core layer, and aggregation layer adopt the modular design so that capacities of these devices can be expanded flexibly with the development of the DC network. The scalability of functions enables the DC to support value-added services. The DC provides functions such as load balancing, dynamic content replication, and VLAN to support value-added service expansion.

1.2.3 Manageability

A manageable network is the prerequisite for successful operation of the DC. The DC provides:

1. Various optimized manageable information
2. Complete QoS functions
3. Integrated SLA management system
4. Capability to manage devices of different vendor
5. Independent background management platform for the DC and users to manage the Network

1.2.4 Security

As a concern of DC users especially e-commerce users, security is a key factor during DC construction. DC security is ensured by security control for the physical space and network. The DC provides an integrated security policy control system to ensure DC security.

1.3 DC Network Architecture

The DC network design is based on the following principles:

- Modular architecture
  The network is deployed in modular architecture that can expand for service adjustment and development.
- High reliability
  The network implements redundant backup of key devices and links. Highly reliable key devices are made up of hot swappable boards and modules, and support redundancy of control modules and power supplies. Network layers are reduced to simplify network architecture and enhance networking reliability.
- Secure isolation
  The DC network adopts effective security control policies that logically isolate data based on services and rights, and uses physical isolation

methods to ensure security of important service data. Services such as server-centered services, IP storage and backup services, and management services are isolated logically. The management network is isolated from other networks physically.

- Manageability and maintainability
  The network is highly manageable. To facilitate maintenance, use integrated products with universal modules.

- Core network area
  This area is the core of the DC network, and connects the inner server area, enterprise intranet, partner enterprise network, disaster recovery center, and external user network.

- Server area

  Servers and application systems are deployed in this area. Based on security and scalability, the server area is divided into the production service area, office service area, testing service area, and the demilitarized zone (DMZ) area and other service areas.

- Interconnection area
  In this area, internal and external enterprise users are connected to the DC. Based on security and scalability, the Internet area is divided into the enterprise intranet, enterprise extranet, and the Internet.

- Disaster recovery center Internet area
  In this area, the disaster recovery centers in the same city are interconnected bytransmission devices and disaster recovery centers in different cities are interconnectedby the WAN leased line.

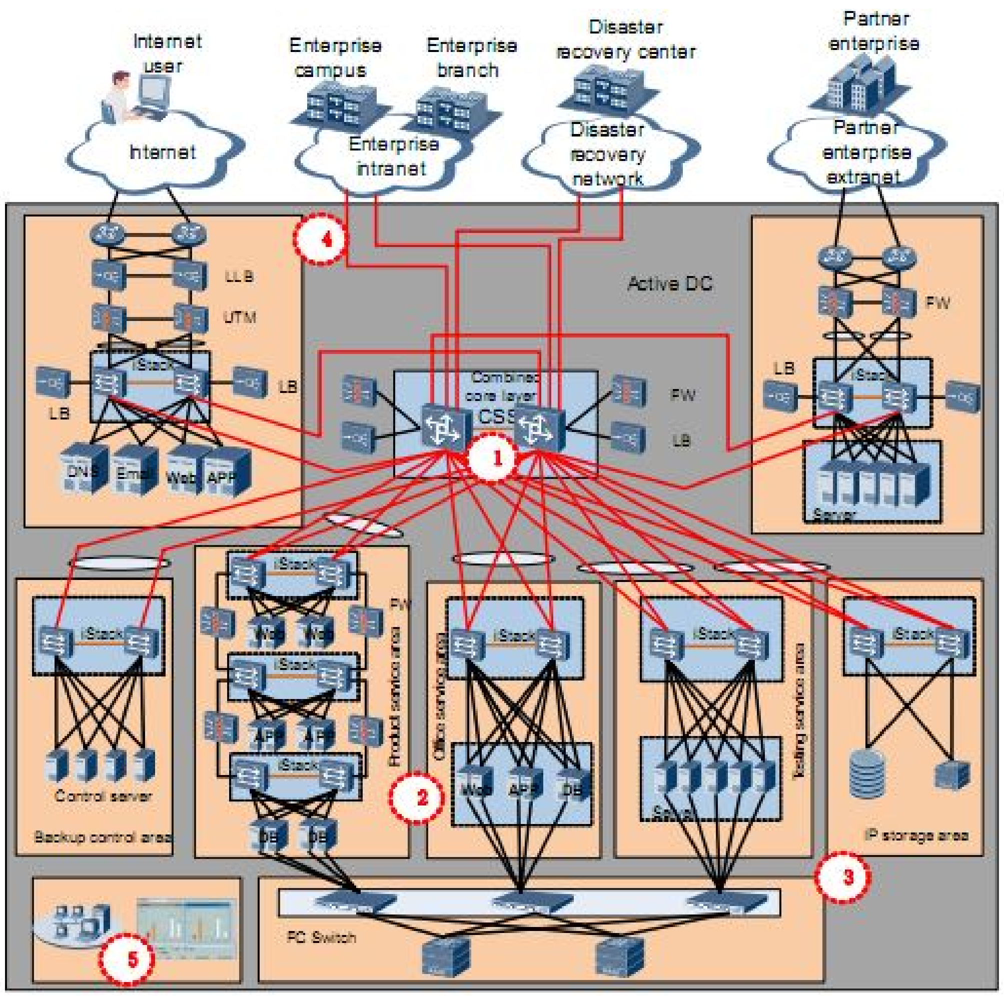


Figure 1.1: Physical Network Architecture

1.4 Thesis Motivation

Data center networks are well structured and layered to achieve high bandwidth and low latency and the bu?er size of ToR (top-of-rack) Ethernet switches is usually small. TCP incast congestion happens in high-bandwidth and low-latency networks, when multiple synchronized servers send data to a same receiver in parallel. DCTCP leverages Explicit Congestion Noti?cation (ECN) and a simple multi-bit feedback mechanism at the host.
There have been some serious concerns about the TCP performance in data center networks, including the long completion time of short TCP ?ows in competition with long TCP ?ows, and the congestion due to TCP incast. Explicit Congestion Noti?cation (ECN) at the intermediate switches can alleviate both problems. The solution - a switch modi?cation only scheme de-queue marking, for further tuning the instant queue length based ECN to achieve optimal incast performance and long ?ow throughput with a single threshold value. In this paper they proposed that using De-queue scheme if the threshold value is properly tuned then without end host modification we can maintain TCP performance.
To address TCP performance issues with a simple ECN tuning at the switch-side only. First, have observed that the network latencies across data center networks are extremely low (under a millisecond). This has inspired to the use of instant queue length instead of average queue length in ECN. We hypothesize that, if instant queue length is used to trigger ECN, and if the threshold is properly tuned, ECN may actually alleviate the TCP performance impact without requiring end-host modifications. We use ECN* in this paper to denote such a scheme in which instant queue length based ECN is used at the switches and standard ECN congestion control is used at the end hosts.
TCP Incast, also known as TCP throughput collapse, is a term used to describe a link capacity under-utilization phenomenon in certain many-to-one communication patterns. Congestion control algorithms have been developed to reduce or eliminate packet drops at the congestion switch. QCN can effectively control link rates very rapidly in a datacenter environment. However, it performs poorly when TCP Incast is observed. Here CP (congestion point) in order to deal with rate fluctuation of different flows within one synchronous reading request. Also use Fair Quantized Congestion Notification (FQCN) to improve fairness of multiple flows.

1.5 Thesis Goal

TCP incast, Queue buildup, Buffer pressure occurs different problems in Data Center Network. It occurs some serious concerns about the TCP performance in data center networks, including the long completion time of short TCP ?ows in competition with long TCP flows and the congestion.
The main goal of our these are as follows:

- Almost full Congestion Control.
- Achieving absolute Fairness.

1.6 Thesis Organization

The paper is organized as follows. In section II, overview of the problem statement of Datacenter Network that causes serious problems. In section III, we describe the related work of the problem statement. We describe in details the whole related work, proposal, contribution, implementation, simulation works and result. In section IV, we describe our proposed technology of QCN in details. In section V, we show our implementation and performance evaluation of our paper work. We describe our simulation tool and performance measurement. In section VI, we describe overall result of our paperwork. In section VII, we describe our future plan and conclusion in details.

Chapter 2

Problem Statements

2.1 Switches

Like most commodity switches, the switches in these clusters are shared memory switches that aim to exploit statistical multiplexing gain through use of logically common packet buffers available to all switch ports. Packets arriving on an interface are stored into a high speed multi-ported memory shared by all the interfaces. Memory from the shared pool is dynamically allocated to a packet by a MMU. The MMU attempts to give each interface as much memory as it needs while preventing unfairness by dynamically adjusting the maximum amount of memory any one interface can take. If a packet must be queued for an outgoing interface, but the interface has hit its maximum memory allocation or the shared pool itself is depleted, then the packet is dropped. Building large multi-ported memories is very expensive, so most cheap switches are shallow buffered, with packet buffer being the scarcest resource.

2.2 TCP Incast

There have been some serious concerns about the TCP performance in data center networks, including the long completion time of short TCP flows in competition with long TCP flows, and the congestion due to TCP incast. Data center networks are designed to support various applications and diverse traffic patterns. Advanced topologies and structures have been proposed to achieve higher bandwidth in data center networks. However, there have been some serious concerns about the TCP performance in data centers. For example, TCP incast congestion has become a practical issue in data center networks. TCP incast occurs when many-to-one short flows are barrier synchronized by the applications. A recent study shows that extreme high bandwidth and low latency, true of most data center networks, are preconditions for incast congestion. Moreover, without well-recognized service differentiation support, short-duration TCP flows will further suffer in the interaction with long-duration TCP flows. Short flows require low queue occupancy for smaller task completion time while a large queue has been built up by long flows.

TCP incast, which results in gross under-utilization of link capacity, occurs in synchronized many-to-one communication patterns. In such a communication pattern, a receiver issues data requests to multiple senders. The senders respond to the request and return an amount of data to the receiver. The data from all senders pass through a bottleneck link in a many-to one pattern to the receiver. When the number of synchronized senders increases, throughput observed at the receiver drops to one or two orders of magnitude below the link capacity. Several factors, including high bandwidth and low latency of a data center network, lead to TCP incast. The barrier synchronized many-to-one communication pattern triggers multiple servers to transmit packets to a receiver concurrently,resulting in a potentially large amount of traffic simultaneously poured into the network. Due to limited buffer space at the switches, such traffic can overload the buffer, resulting in packet losses. TCP recovers from most of packet losses through timeout retransmission. The timeout duration is at least hundreds of milliseconds, which is orders of magnitude greater than a typical round trip time in a data center network. A server that suffers timeout is stalled even though other servers can use the available bandwidth to complete transmitting. TCP recovers from most of packet losses through timeout retransmission. The timeout duration is at least hundreds of milliseconds, which is orders of magnitude greater than a typical round trip time in a data center network. A server that suffers timeout is stalled even though other servers can use the available bandwidth to complete transmitting. Due to the synchronized communication, however, the receiver has to wait for the slowest server that suffers timeout. During such awaiting period, the bottleneck link may be fully idle. This results in underutilization of the link and performance collapse.

2.3 Queue buildup

Long-lived, greedy TCP flows will cause the length of the bottleneck queue to grow until packets are dropped, resulting in the familiar saw tooth pattern when long and short flows traverse the same queue, two impairments occur. First, packet loss on the short flows can cause incast problems as described above. Second, there is queue buildup impairment: even when no packets are lost, the short flows experience increased latency as they are in queue behind packets from the large flows. Every worker in the cluster is handling both query traffic and background traffic (large flows needed to update the data on the workers), so this traffic pattern occurs very frequently.

2.4 Buffer pressure

Given the mix of long and short flows in data centers as exemplified by the application we studied, it is very common for short flows on one port to be impacted by activity on any of the many other ports, as depicted. Surprisingly, the loss rate of short flows in this traffic pattern depends on the number of long flows traversing other ports. The explanation is that activity on the different ports is coupled by the shared memory pool. The long, greedy TCP flows build up queues on their in interfaces. Since buffer space is a shared resource; the queue build up reduces the amount of buffer space available to absorb bursts of traffic from Partition/Aggregate traffic. We term this impairment buffer pressure. The result is packet loss and timeouts, as in incast, but without requiring synchronized flows.

Chapter 3

Related Work

In recent years, data centers have transformed computing, with large scale consolidation of enterprise IT into data center hubs, and with the emergence of cloud computing service providers like Amazon, Microsoft and Google. A consistent theme in data center design has been to build highly available, highly performance computing and storage infrastructure using low cost, commodity components. Datacenters are becoming increasingly important to provide a myriad of services and applications to store, access, and process data. Datacenters are typically composed of storage devices, servers and Ethernet switches to interconnect servers within datacenters. Data may be spread across or striped across many servers for performance or reliability reasons. During data access from servers, the data need to pass through datacenter Ethernet switches. The Ethernet switches have small buffers in the range of 32-256 KB typically, which may be over?owed at congestions. Due to small buffer it causes some serious problem like TCP incast, Queue buildup, Buffer pressure etc. To minimize these problems several works have been done.

3.1 DCTCP:

Data center networks are well structured and layered to achieve high bandwidth and low latency and the bu?er size of ToR (top-of-rack) Ethernet switches is usually small. TCP incast congestion happens in high-bandwidth and low-latency networks, when multiple synchronized servers send data to a same receiver in parallel. DCTCP leverages Explicit Congestion Noti?cation (ECN) and a simple multi-bit feedback mechanism at the host. In the data

center, operating with commodity, shallow buffered switches, DCTCP delivers the same or better throughput than TCP, while using 90
In this paper, we make two major contributions:

1. Measure and analyze production data center traf?c that uses commodity switches. Impairments that hurt performance are identi?ed, and linked to properties of the traf?c and the switches.

2. We propose a TCP variant, DCTCP, which addresses these impairments to meet the needs of applications.

3.1.1 Understanding Performance Impairments

- Switches
  Like most commodity switches, the switches in these clusters are shared memory switches that aim to exploit statistical multiplexing gain through use of logically common packet buffers available to all switch ports. Packets arriving on an interface are stored into a high speed multi-ported memory shared by all the interfaces. Memory from the shared pool is dynamically allocated to a packet by a MMU. The MMU attempts to give each interface as much memory as it needs while preventing unfairness by dynamically adjusting the maximum amount of memory any one interface can take. If a packet must be queued for an outgoing interface, but the interface has hit its maximum memory allocation or the shared pool itself is depleted, then the packet is dropped. Building large multi-ported memories is very expensive, so most cheap switches are shallow buffered, with packet buffer being the scarcest resource.

- Incast
  If many flows converge on the same interface of a switch over a short period of time, the packets may exhaust either the switch memory or the maxi-mum permitted buffer for that interface, resulting in packet losses for some of the flows. Degrading both performance and, more importantly, user experience. The problem is that a response that incurs incast will almost certainly misses the aggregator deadline and be left out of the final results.

- Queue buildup
  Even when no packets are lost, the short flows experience increased latency as they are in queue behind packets from the large flows.

3.1.2 DCTCP algorithm

1. Simple Marking at the Switch
   DCTCP employs avery simple active queue management scheme. There is only a single parameter, the marking threshold, K. An arriving packet is marked with the CE code point if the queue occupancy is greater than K upon its arrival. Otherwise, it is not marked. The design of the DCTCP marking scheme is motivated by the need to min Other variants which use a variety of fixed factors and/or other fixed reactions have the same issue.

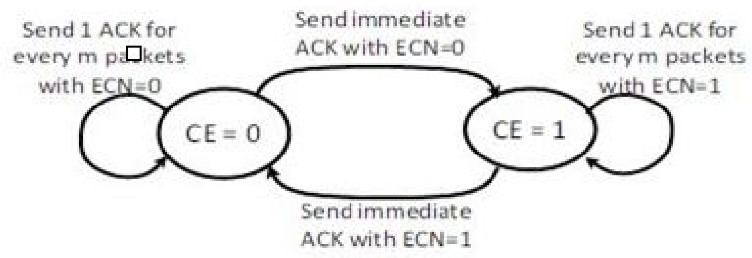
   
   Send 1 ACK for every m pa□kets with ECN=0
Send immediate ACK with ECN=0
Send 1 ACK for every m packets with ECN=1
CE = 0
CE = 1
Send immediate ACK with ECN=1

   Figure 3.1: Two state ACK generation state machine

   DCTCP aggressively marks packets when a queue overshoot is sensed. This allows sources to be notified of the queue overshoot as fast as possible. Figure 10 shows how the RED marking scheme (implemented by most modern switches) can be re-purposed for DCTCP. We simply need to set both the low and high thresholds to K, and mark based on instantaneous, instead of average queue length.

2. ECN-Echo at the Receiver The only difference between a DCTCP receiver and a TCP receiver is the way information in the CE code points is conveyed back to the sender. RFC 3168 states that a receiver sets the ECN-Echo flag in a series of ACK packets until it receives confirmation from the sender (through the CWR flag) that the congestion notification has been received. A DCTCP receiver, however, tries to accurately convey the exact sequence of marked packets back to the sender. The simplest way to do this is to ACK every packet, setting the ECN-Echo flag if and only if the packet has a marked CE code point. However, using Delayed ACKs is important for a variety of reasons, including reducing the load on the data sender.
   To use delayed ACKs (one cumulative ACK for every m consecutively

received packets 3), the DCTCP receiver uses the trivial two state state-machines shown in Figure 11 to determine whether to set ECN-Echo bit. The states correspond to whether the last received packet was marked with the CE code point or not. Since the sender knows how many packets each ACK covers, it can exactly reconstruct the runs of Marks seen by the receiver

3. Controller at the Sender
The sender maintains a running estimate of the fraction of packets that are marked,which is updated once for every window of data (roughly one RTT) as follows:

$$\alpha \leftarrow (1 - g) \times \alpha + g \times F$$

3.2 Tuning ECN for Data Center Networks

There have been some serious concerns about the TCP performance in data center networks, including the long completion time of short TCP flows in competition with long TCP flows, and the congestion due to TCP incast. In this paper, we show that a properly tuned instant queue length based Explicit Congestion Notification (ECN) at the intermediate switches can alleviate both problems. Compared with previous work, our approach is appealing as it can be supported on current commodity switches with a simple parameter setting and it does not need any modification on ECN protocol at the end servers. Furthermore, we have observed a dilemma in which a higher ECN threshold leads to higher throughput for long flows whereas a lower threshold leads to more senders on incast under buffer pressure. We address this problem with a switch modification only scheme - dequeue marking, for further tuning the instant queue length based ECN to achieve optimal incast performance and long flow throughput with a single threshold value

- ECN :
As an explicit congestion notification protocol, DECbit is proposed to detect possible congestion at routers. A bit at data packet header is set by intermediate nodes to notify the congestion. When the receiver gets the marked packets, it echoes the information to the sender by marking another bit at the ACK. The sender then decides whether the congestion window should be cut by the ratio of markedACKs. Recent switches/routers follow the standard set by RFC in implementing

ECN for TCP, which is closely related to Random Early Detection (RED) . RED addresses the global synchronization issue of multiple TCP connections sharing the same bottleneck link, which is caused by a large number of packet losses occurring when the drop tail buffer overflows. For ECN/RED, an average queue length is maintained at the switch, and two (low and high) thresholds for the average queue length are used. In RED, when the average queue length is between the two thresholds, the incoming packet is dropped with a probability capped by a parameter, namely the max drop probability. When the average queue length is over the high threshold, all incoming packets are dropped. In ECN, the incoming packets will be marked on Congestion Experienced (CE) bits instead of being dropped. In contrast to RED, which works without changing TCP end points, ECN requires a TCP receiver to continuously pass the CE bits as ECE (ECN echo) bits to the TCP sender. The TCP sender then cuts the congestion window in half and marks the first new data with a CWR (congestion window reduced) bit to suppress the ECE bit from the receiver.

- ECN* :
  As already mentioned, instant queue length based ECN has been proposed in previous research to decouple ECN from AQM [15] and for frequent marking of the congestion state. Here, we look at the instant queue length from a more generalized angle. Instant queue length represents the congestion window of all TCP connections sharing the same bottleneck. If the focus is to deal with temporal congestion caused by traffic burstiness, incast congestion, a congestion control scheme like ECN can use instant queue length directly. Therefore, as long as the ECN threshold is set according to the window reduction taken by TCP, the TCP throughput performance will not be degraded. Our generalized ECN strategy works as follows. At the intermediate switch/router, the instant queue length value is compared with a preconfigured threshold value whenever a packet is processed. If the instant queue length is greater Than or equal to the threshold, the packet is marked with a CE bit at the IP header. The only configurable parameter that can tune the behavior of this scheme is the ECN threshold. Our generalized ECN strategy does not make any assumption on the congestion control scheme used by the TCP sender and receiver to handle the marked packets, but we believe different schemes should achieve similar performance and have a similar tradeoff on parameter settings. The congestion control schemes include DCTCP, standard ECN as defined in RFC3168, and DECbit. We designate ECN* as the scheme

that uses instant queue length and a single threshold at the switches and uses standard ECN at the TCP sender/receiver. ECN* is very well supported by today's commodity ECN capable switches. First, ECN switches allow a weight parameter to adjust the exponential factor for updating the average queue length. By setting this weight to 1, the average queue length is effectively the same as the instant queue length because the values in the history are ignored. Second, ECN switches accept two threshold parameters, the low and high thresholds. By setting the two thresholds to the same value, they become one single threshold and the region in between the low and high thresholds is no longer in effect. The support of ECN* at end-servers follows the ECN standard.

- Problem statement :
  To understand the impact of the ECN threshold setting on performance, we categorize the traffic in data center networks into three categories as follows. First, all competing TCP flows are long flows. In this case, the performance is mainly determined by the throughput of the long flows. A recent study in showed that there are usually 2 or 3 competing long flows. Second, a few short flows are trying to deliver a small volume of data as fast as possible, while there are other background long flows competing for bandwidth. In this case, the completion time of those short flows is the focus of performance. Third, during TCP incast, a large number of short flows are competing for a specific network port in a very short time. TCP incast happens whether or not there are ongoing background long flows. The performance of TCP incast is determined by the completion time of the last finished short flow.
  As shown in Section 3.4, TCP (ECN off) actually works well for the first category. However, the large buffer occupation of TCP (ECN off) makes it work poorly for the second and third cases. ECN* and DCTCP use congestion control based on instant queue length. Thus, compared with TCP, both methods achieve similar performance for the first category and much better performance for the second and third categories. Considering the throughput of TCP connections is still fundamental to performance, we need a scheme that improves the performance of both DCTCP and ECN* during the last two categories by assuming the threshold is set for the first category. In Section 3.2, we present the relationship of throughput and the threshold setting of ECN*. For DCTCP, the relationship has also been analyzed previously in. Given that the ECN threshold has been set, we must next address how to make the instant queue length represent the traffic and thus trigger the

threshold faster. Note that we don't wish to change the ECN threshold dynamically according to the traffic categories as we believe that such a solution is hard to implement due to the traffic dynamics in a data center.

- Dequeue marking scheme :
  We have proposed, implemented, and evaluated a pure switch based solution - dequeue marking. The purpose of dequeue marking is not only to provide a complete switch only solution for ECN*, but also to understand the performance limit of instant queue length based ECN (both ECN* and DCTCP). Customers that have concerns with TCP stack modifications at the end server, e.g., DCTCP, can use dequeue marking and ECN*.
  At the concept level, dequeue marking seems similar to the well-known drop-from-front method , which was proposed for TCP over ATM. A previous study in [16] shows that drop-from-front greatly improves performance over tail drop, but for RED, front-drop and tail-drop are similar. In existing commodity switches, ECN follows the implementation of RED. As the switch buffer uses a First Come First Service (FIFO) rule, RED checks whether a packet should be dropped (or marked if ECN is enabled) when an incoming packet is queued at the switch output port. Such a dropping/ marking policy for original RED/ECN works well as the rule is performed by an exponential filter based on the average queue length. In this paper, we argue that for instant queue length based ECN, the marking policy performed when packets are queued is no longer efficient. For instant queue length based ECN, e.g., ECN*, the performance has been analyzed by assuming that congestion information is generated when the instant queue length is over the threshold. However, in existing ECN implementation, such congestion information (marked CE bit on packets just queued) must wait until the marked packet moves to the head of the queue. If the ECN threshold is set to a large value to accommodate TCP throughput of long flows, we believe that setting ECN mark when packets are queued severely delays the delivery of congestion information. In this paper, we propose the use of dequeue marking for instant queue length based ECN at switches. When an ECN-capable packet is about to be dequeued, we check the instant queue length and the ECN threshold. If the instant queue length is larger or equal to the ECN threshold, then the packet is marked with a CE bit. There are two benefits to dequeue marking. First, the latency to deliver the congestion information is reduced, so a better incast performance is expected. Second, our analysis and exper-

imental results (skipped due to space limitation) for dequeue marking show a minimal buffer size (threshold upper bound) of ECN* . Dequeue marking decides whether a packet should be marked when a packet is about to transmit, which is different from a straightforward extension of drop-from-front strategy of RED to a mark-from-front strategy of ECN. This is because for both RED and ECN, the dropping/marking decision is made when a packet is enqueued. Considering that the traffic is highly bursty during TCP incast and multiple packets are enqueued at the same time, the front-mark of ECN may only mark the packets waiting for transmission when other packets are enquired, while dequeue marking continuously marks all outgoing packets until the queue length is less than the threshold. To this end, front-mark may leave some "holes" (unmarked packets) and thus we believe dequeue marking is more suitable for instant queue based ECN.

- Interaction of long flows and short flows :
  Dequeue marking is proposed to address the ECN threshold setting problem for incast. Readers may have concerns over whether such modifications will make short flows unnecessarily conservative when competing with long flows that have a larger congestion window and queue occupation at the switches. To evaluate the interaction between long and short flows, we first established two long connections to the same receiver that occupy the buffer on the bottleneck link. Then we continuously established and tore down a new TCP connection to transmit 20kbytes data to the same destination server of the two long flows. The duration of the long connections was just long enough to cover the transmission of all the 20kbytes short connections. Both long and short TCP connections were started using iperf. All the servers were under the same GbE switch. We selected 20kbytes as it has been used in related research. We evaluated the performance according to two aspects of the interaction: the completion time observed by short flows and the throughput obtained by long flows. In Figure 10 and Figure 11, we show the distribution of completion time of short flows. Note that the completion time of TCP (ECN off) was actually worse and had over a 2

3.3 On Mitigating TCP incast in DCN

TCP Incast, also known as TCP throughput collapse, is a term used to describe a link capacity under-utilization phenomenon in certain many-to-one communication patterns, typically in many datacenter applications. The main root cause of TCP Incast analyzed by prior works is attributed to packet drops at the congestion switch that result in TCP timeout. Congestion control algorithms have been developed to reduce or eliminate packet drops at the congestion switch. In this paper, the performance of Quantized Congestion Notification (QCN) with respect to the TCP incast problem during data access from clustered servers in datacenters are investigated. QCN can effectively control link rates very rapidly in a datacenter environment. However, it performs poorly when TCP Incast is observed. To explain this low link utilization, we examine the rate fluctuation of different flows within one synchronous reading request, and find that the poor performance of TCP throughput with QCN is due to the rate unfairness of different flows. Therefore, an enhanced QCN congestion control algorithm, called fair Quantized Congestion Notification (FQCN), is proposed to improve fairness of multiple flows sharing one bottleneck link. We evaluate the performance of FQCN as compared to that of QCN in terms of fairness and convergence with four simultaneous and eight staggered source flows. As compared to QCN, fairness is improved greatly and the queue length at the bottleneck link converges to the equilibrium queue length very fast. The effects of FQCN to TCP throughput collapse are also investigated. Simulation results show that FQCN significantly enhances TCP throughput performance in a TCP Incast setup.

- QCN :
  The QCN algorithm is composed of two parts as shown in Fig. 1a: switch or congestion point (CP) dynamics and rate limiter (RL) or reaction point (RP) dynamics. At CP, the switch buffer attached to an oversubscribed link samples incoming packets and feeds back the congestion severity level back to the source of the sampled packet. While at RP, RL associated with a source decreases its sending rate based on congestion feedback message received from CP, and increases its rate voluntarily to recover lost bandwidth and probe for extra available bandwidth.

- The CP Algorithm
  The goal of CP is to maintain the buffer occupancy at a desired operating point, Qeq. CP samples the incoming packet with a probability depending on the severity of congestion measured by Fb, and computes the severity of congestion measurement Fb. Fig. 1b shows the sampling probability as a function of |Fb|. Fb is calculated as follows:

  $$Fb = -(Qoff + w * Q)$$

where

$$Qoff = Q - Qeq, Q\_ = Q - Qold = Qa - Qd$$

- The RP Algorithm

  The RP algorithm adjusts the sending rate by decreasing the sending rate based on the congestion feedback message received from CP, and increasing the sending rate voluntarily to recover lost bandwidth and to probe for extra available bandwidth. Rate decreases: when a feedback message is received, current rate (CR) and target rate (TR) are updated as follows:

  $$TR = CR$$

  $$CR = CR(1 - Gd|Fb|)$$

  where the constant Gd is chosen to ensure that the sending rate cannot decrease by more than 50

- FQCN:

  One problem with the QCN congestion control algorithm is rate unfairness of different flows when sharing one bottleneck link. Fig. 2 shows the simulation results of the source goodput of different flows for four simultaneous constant bit rate (CBR) source flows with User Datagram Protocol (UDP) traffic. In this simulation, each source node is linked to an edge switch. All edge switches are linked to a single core switch. The capacity of each link is 10 Gbps. The propagation delay of each link is assumed to be 0.5 ?s. The processing delay of each node is assumed to be 1 ?s. Each switch adopts the drop tail queue mechanism. Note that the four sources achieve very different rates by the QCN algorithm, showing the unfairness of QCN. In the above analysis on TCP Incast, the total TCP goodput is limited by the slowest source flow in the TCP Incast problem. Therefore, the unfairness of QCN decreases the TCP throughput. In this section, we propose an enhancement on QCN, referred to as FQCN, to improve TCP throughput to avoid or postpone the instigation of TCP Incast. FQCN feeds QCN messages back to all flow sources which send packets with the sending rate over their share of the bottleneck link capacity. FQCN adopts the same RP algorithm as that of QCN. The main modification is at the congestion switch. The basic algorithm of fair QCN at CP is summarized as follows:

- Almost all FQCN operations are the same as those inQCN.
- The switch monitors the queue length and the packetarrival rate of each flow.
- The switch calculates the main measure of congestionseverity level, Fb. If Fb$<$ 0, the negative QCN feedback

3.4 Quantized Congestion Notification for Multicast in DCN

When convention QCN applied to multicast flows, a sending device might receive congestion feedback from multiple locations and multiple flows may suffer lower throughput than unicast flows going through the same bottlenecks. So they propose QCN/BS the sending device identifies the transmission rate of each bottleneck point and selects the most congested switch so congestion can't happen. In basic QCN algorithm there are two points. They are Congestion point (QCN source) Reaction Point (Switch dynamics). In QCN/BS, the source RP has been modified. Thus simple change can stop being congested.

3.5 An Effective Approach to Preventing TCP Incast Throughput Collapse for Data Center Networks

This paper presents an effective solution to the known TCP incast problem in data center networks. The incast problem refers to a drastic TCP throughput drop when the number of servers synchronically sending data to the same receiver is too large. Our proposed approach utilizes the link bandwidth as fully as possible but without significant packet losses by limiting the number of concurrent senders to a reasonable value. Based on the concept of bandwidth delay product, our approach conservatively estimates the reasonable number of concurrent senders. Our approach does not modify TCP protocol itself and can thus be applied to any TCP variant, and works regardless of the

type of data center network topology and throughput limitation. Analysis and simulation results show that our approach eliminates the incast problem and noticeably improves TCP throughput.
More specifically, a TCP flow is generally referred to as either a TCP mouse if it is short (so short that it never exits the slow start phase), or a TCP elephant if it has a bulk of data to transfer and lasts very long.In this paper, we study TCP flows in data center, which enter both the slow-start and congestion avoidance phases but have only a limited amount of data to transfer at a time so their durations are not as long as normal TCP elephants. Since the TCP window for such a medium-duration flow tends to grow exponentially fast during the slow-start phase ((just like how a rabbit can spring but only for a limited amount of time), we call such a medium-duration TCP flow a TCP "rabbit". For a given limited amount of data required by the receiver, having just one server sending a "rabbit" flow at a time may not fully utilize the bandwidth on the link. On the other hand, due to the limited switch buffer, too many servers send to the receiver simultaneously will lead to overfilling of the buffer, resulting in TCP timeouts and then TCP incast. Accordingly the number of servers that are permitted to send data at the same time should be carefully determined. Our objective is to find a reasonable value of number of servers that are permitted to simultaneously send data to the receiver, denoted by m, which ensures that the aggregate bandwidth is utilized efficiently but without packet losses.

- Model
  TCP incast problem was first observed in cluster-based storage systems. It has become a practice issue with recent research works which show that many applications in data centers use many-to-one communication pattern. The model presented in [1] abstracts the most basic representative setting in which TCP incast occurs. In order to simplify our analysis we use the same model. In such an environment, the receiver requests data blocks from n servers. Each data block is striped across n servers, such that each server stores a chunk of a data block, also referred to as a Server Request Unit (SRU). The receiver sends request to servers for that particular block. Once all servers return their corresponding chunk to it, the receiver will request for the next block. Since the networks atop TCP/IP and Gigabit Ethernet have a high bandwidth (one to ten Gigabit bits per second) and low latency (round-trip-times of ten to a hundred microseconds), the link connecting the receiver to the switch is usually the bottleneck.

- Proposed Approach:

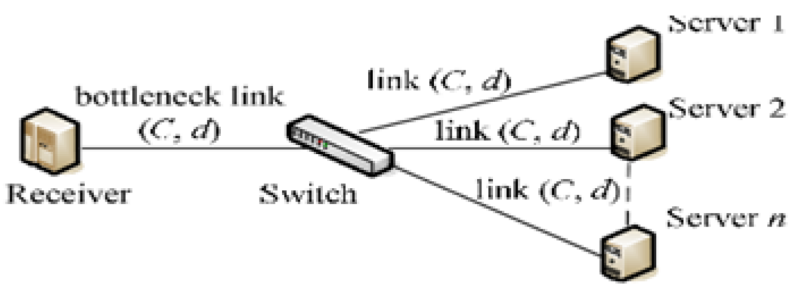


Figure 3.2: A simplified TCP incast model

$$W_s = min(W_c, W_r)$$

If TCP has enough data to sent, then during any RTT, the maximum number of packets on the wire,donated by M, is Ms. In data centers, however, since a TCP is a rabbit, even though in principle it is allowed to have up to Ws packets on the wire, the number of packets actually injected to the network may be less. More specifically, M is limited by

$$M = \min(Ws, Ds)$$

Chapter 4

Proposed Technology

The QCN algorithm is based on a congestion notification from a switch. A QCN switch observes current queue length and calculate the feedback value when a data frame is arrives. When congestion occurs, as identified by the calculated feedback value ,the switch sends a feedback frame with a certain probability to the source. To work with the QCN algorithm we need two modifications:

1. Server End modification.
2. Switch End modification.

- Switch End Modification:
  We first have to check the buffer size, B and number of servers connected to the switch, m and queue occupancy, q. Then we have to calculate the threshold value that generate negative (-ve) feedback.
  So, Threshold to generate -ve feedback, b= B-m.
  If q>= b then generate negative feedback.
  If q<b then generate positive feedback.
  This feedback message will be broadcast to all servers connected to the switch.

- Server End Modification:
  When received positive feedback - Start Transmission.
  When received negative feedback - Stop transmission.
  For rate control, we followed QCN mechanism:

  1. Rate decrease:
     Whenever a feedback frame is received, the RP first active the

rate decrease phase and both the byte counter and time counter are initialized reset to 0. Upon receiving negative feedback, the server stops the transmission. And the rate decrease reduces as follows:

$$TR \leftarrow CR$$

$$CR \leftarrow 0.5 * CR$$

2. Recovery:
   After rate decreasing if no further negative feedback is received then the rate will be recovered as follows:
   - Fast recovery:

     In the fast recovery phase, CR is increased rapidly just after the rate reduction if no feedback is received. When a time counter or byte counter is incremented by 1.After first rate decrease the CR will be changed as follows:

     $$\mathbf{CR \leftarrow 0.5 * (CR + TR)}$$

     And the byte counter will be initialized.
   - Active Increase:

     When the Byte counter reaches 5 and yet no negative feedback is received then it enters to Active increase state. And TR and CR changed as follows:

     $$\mathbf{TR \leftarrow TR + R_{AI}}$$

     $$\mathbf{CR \leftarrow 0.5 * (CR + TR)}$$

Chapter 5

Implementation

In implementation we evaluate the performance of QCN. We used OMNET++ for performance evaluation.OMNeT++ is an object-oriented modular discrete event network simulation framework. It has a generic architecture, so it can be (and has been) used in various problem domains.OMNeT++ itself is not a simulator of anything concrete, but rather provides infrastructure and tools for writing simulations. One of the fundamental ingredients of this infrastructure is a component architecture for simulation models.
Before implementation we took some assumptions:

- 3 Switch is ideal,routing the packets don't take time and routing tables are already configured.
- No delay and errors on channels.
- Host doesn't create packets faster than the current rate.
- Packets are normal distributed.
- Feedback messages have priority over regular messages.

5.1 Omnet++ simulator

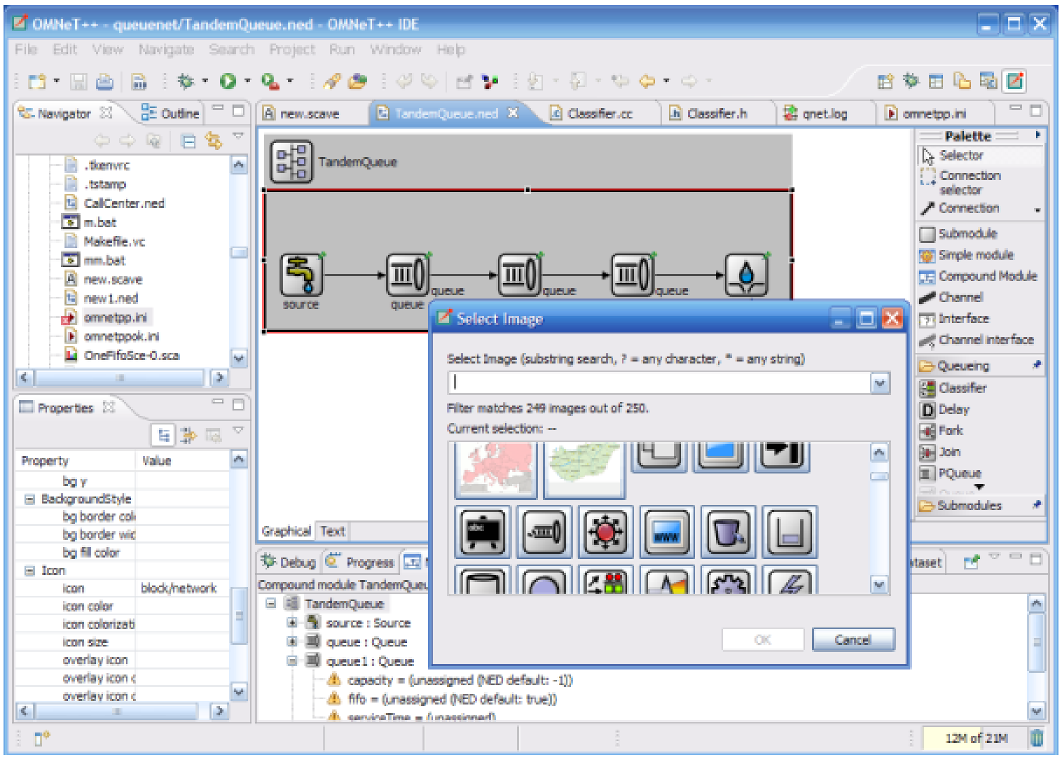
OMNeT++ is an extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators. Network in this case is meant in a broader sense that includes wired and wireless communication networks, on-chip networks, queuing networks, OMNeT++ provides component architecture for models. Components (modules) are programmed

in C++, then assembled into larger components and models using a high-level language (NED)
OMNeT++ is not a simulator in itself but rather a simulation framework. Instead of containing explicit and hardwired support for computer networks or other areas, it provides the infrastructure for writing such simulations. Specific application areas are catered by various simulation models and frameworks, most of them are open source. These models are developed completely independently of OMNeT++, and follow their own release cycles.
OMNeT++ provides a component architecture for models. Components (modules) are programmed in C++, then assembled into larger components and models using a high-level language (NED). Reusability of models comes for free. OMNeT++ has extensive GUI support, and due to its modular architecture, the simulation kernel (and models) can be embedded easily into your applications.

- Components :
  - Simulation kernel library.
  - NED topology description language.
  - OMNeT++ IDE based on the Eclipse platform.
  - GUI for simulation execution, links into simulation executable (Tkenv)
  - Utilities (makefile creation tool, etc.)
  - Documentation, sample simulations, etc.
  - Command-line user interface for simulation execution (Cmdenv)
- Platforms :
  OMNeT++ runs on Windows, Linux, Mac OS X, and other Unix-like systems. The OMNeT++ IDE requires Windows, Linux or Mac OS X.

5.2 Screenshots of different components of the IDE:

1. Graphical ned editor :
   After configuring the code we got the formal result of QCN which derives the theperformationevalution of QCN.Then we ran the simulation code and get the result.



Select Image (substring search, ? = any character, * = any string)

Filter matches 249 images out of 250.

Current selection: --

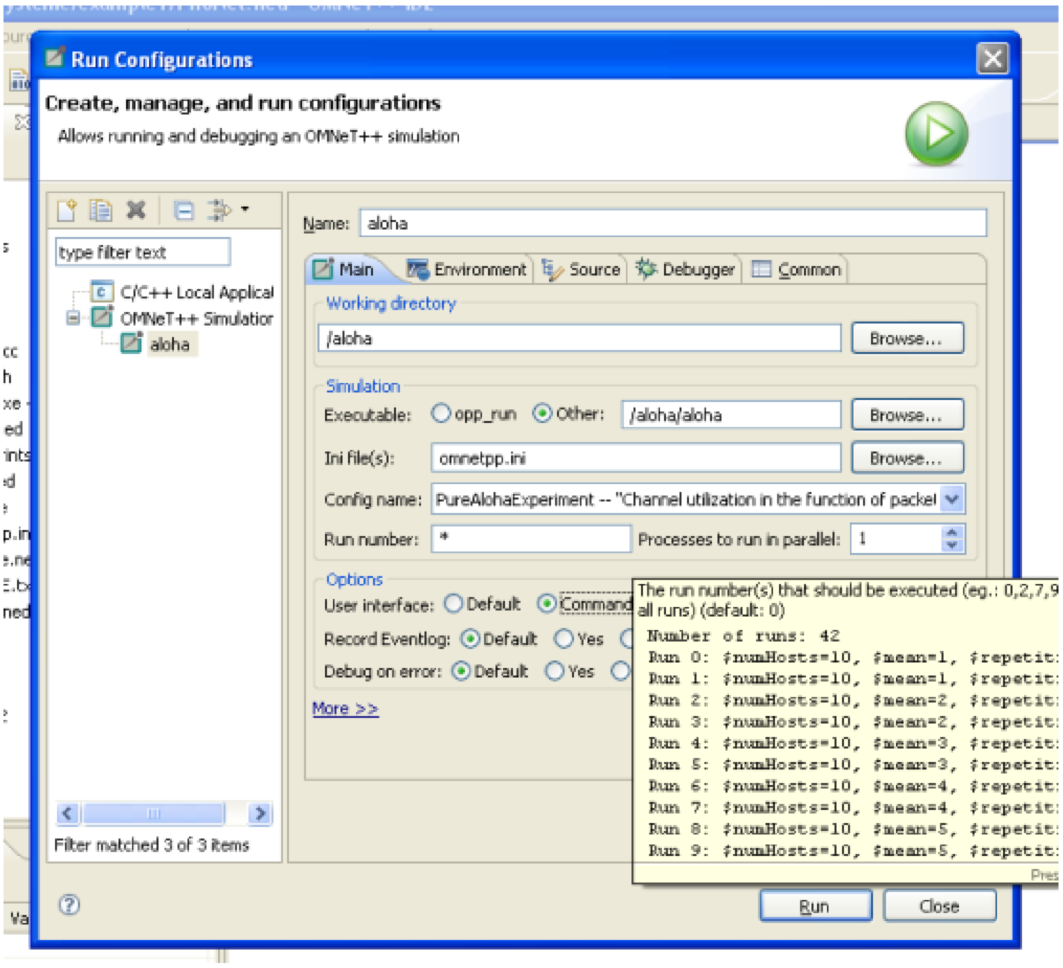
Property	Value
bg y	
⊟ BackgroundStyle	
bg border col	
bg border wid	
bg fill color	
⊟ Icon	
icon	block/network
icon color	
icon colorizati	
icon size	
overlay icon	
overlay icon c	
overlay icon c	

Compound module TandemQueu

```
⊟ TandemQueue
   ⊞ source : Source
   ⊞ queue : Queue
   ⊟ queue1 : Queue
       capacity = (unassigned (NED default: -1))
       fifo = (unassigned (NED default: true))
       serviceTime = (unassigned)
```

Figure 5.1: Graphical ned editor

2. Launching a simulation :
   Our test application performs synchronized reads over TCP in OMNET ++ to model a typical striped ?le system data transfer operation to test the effects of QCN on the TCP Incast problem. The TCP throughput collapse for a synchronized reading application performed on the network with the congestion control algorithm as follows



Name: aloha

Working directory: /aloha

Simulation
Executable: opp_run [✓ Other:] /aloha/aloha
Ini file(s): omnetpp.ini
Config name: PureAlohaExperiment -- "Channel utilization in the function of packet
Run number: *
Processes to run in parallel: 1

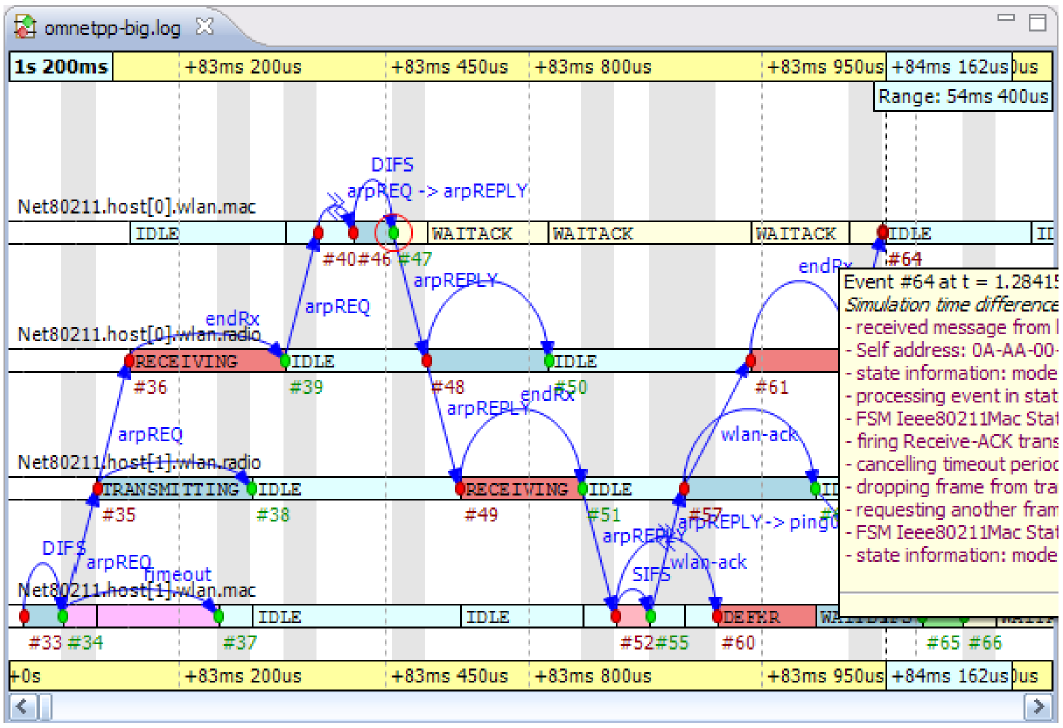
Options
User interface: Default [✓ Command]
Record Eventlog: [✓ Default] Yes
Debug on error: [✓ Default] Yes

The run number(s) that should be executed (eg.: 0,2,7,9
all runs) (default: 0)

```
Number of runs: 42
Run 0: $numHosts=10, $mean=1, $repetit
Run 1: $numHosts=10, $mean=1, $repetit
Run 2: $numHosts=10, $mean=2, $repetit
Run 3: $numHosts=10, $mean=2, $repetit
Run 4: $numHosts=10, $mean=3, $repetit
Run 5: $numHosts=10, $mean=3, $repetit
Run 6: $numHosts=10, $mean=4, $repetit
Run 7: $numHosts=10, $mean=4, $repetit
Run 8: $numHosts=10, $mean=5, $repetit
Run 9: $numHosts=10, $mean=5, $repetit
```

Figure 5.2: Launching a simulation

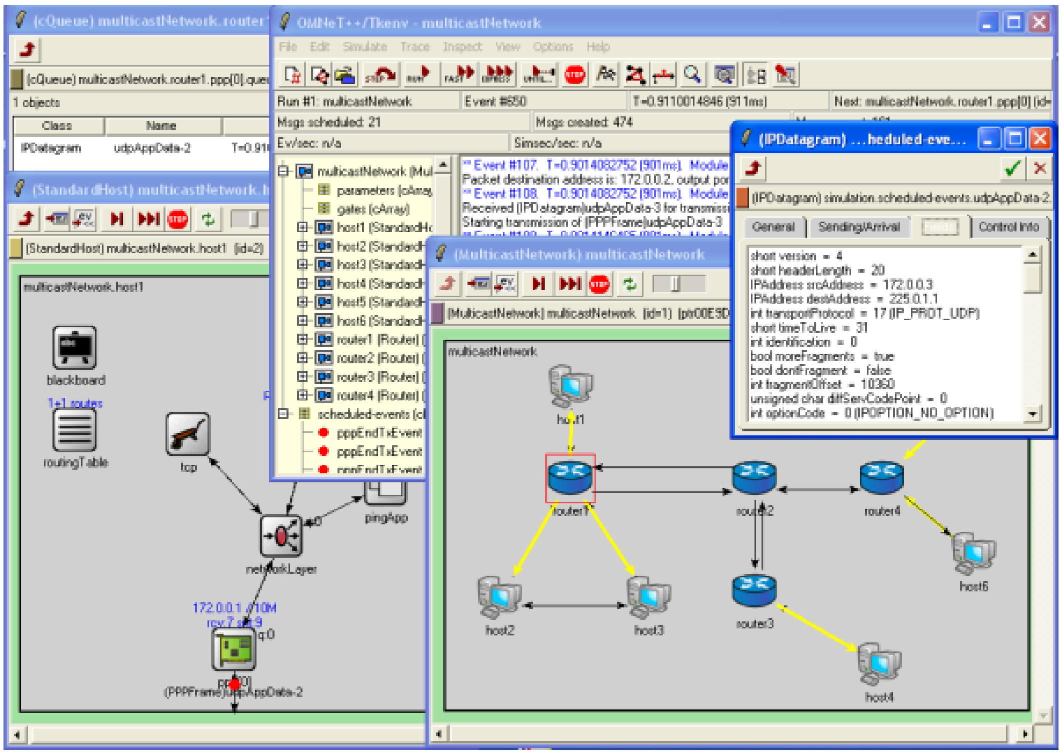
3. Sequence chart :
   QCN can effectively control link rates very rapidly in a datacenter environment.We used different eventlog to evaluate the performance of the simulation.



Event #64 at t = 1.28415
Simulation time difference
- received message from I
- Self address: 0A-AA-00-
- state information: mode
- processing event in stat
- FSM Ieee80211Mac Stat
- firing Receive-ACK trans
- cancelling timeout perioc
- dropping frame from tra
- requesting another fram
- FSM Ieee80211Mac Stat
- state information: mode

Figure 5.3: Sequence chart

4. Graphical runtime environment :
   Finally we get the following result.The result shows that



```
Run #1: multicastNetwork    Event #650    T=0.9110014846 (911ms)    Next: multicastNetwork.router1.ppp[0] (id=
Msgs scheduled: 21    Msgs created: 474
Ev/sec: n/a    Simsec/sec: n/a

** Event #107.  T=0.9014082752 (901ms).  Module
Packet destination address is: 172.0.0.2, output por
** Event #108.  T=0.9014082752 (901ms).  Module
Received (IPDatagram)udpAppData-3 for transmissi
Starting transmission of (PPPFrame)udpAppData-3
```

```
short version = 4
short headerLength = 20
IPAddress srcAddress = 172.0.0.3
IPAddress destAddress = 225.0.1.1
int transportProtocol = 17 (IP_PROT_UDP)
short timeToLive = 31
int identification = 0
bool moreFragments = true
bool dontFragment = false
int fragmentOffset = 10360
unsigned char diffServCodePoint = 0
int optionCode = 0 (IPOPTION_NO_OPTION)
```

Figure 5.4: Graphical runtime environment

Chapter 6

Result

Datacenters are becoming increasingly important to provide a myriad of services and applications to store, access and process data.
The result of our research are as follows:

- Analyze a network with high data Rates, and a congested links.
- Activate the QCN algorithm in such a network and analyze the RTT times and stability of the network .
- Compare the behavior of congested link with and without the QCN algorithm.
- Compare several parameters in the QCN algorithm itself.

Chapter 7

Conclusion

Datacenters are becoming increasingly important to provide a myriad of services and applications to store, access, and process data. Datacenters are typically composed of storage devices, servers and Ethernet switches to interconnect servers within datacenters. Our analysis reveals that QCN work well with small buffer sizes in a huge data center. From our analysis we can achieve almost 100

7.1 Thesis Summary

The QCN algorithm is based on a congestion notification from a switch. A QCN switch observes current queue length and calculates the feedback value when a data frame is arrives. When congestion occurs, as identified by the calculated feedback value, the switch sends a feedback frame with a certain probability to the source. To work with the QCN algorithm we need two modifications:

1. Server End modification.
2. Switch End modification.

We first have to check the buffer size, B and number of servers connected to the switch, m and queue occupancy, q. Then we have to calculate the threshold value that generate negative (-ve) feedback.
So, Threshold to generate -ve feedback, $b = B - m$.
If $q >= b$ then generate negative feedback.
If $q < b$ then generate positive feedback.

This feedback message will be broadcast to all servers connected to the switch.
For server end modification, we use different rate algorithm to maintain the length.

7.2 Contribution

Data may be spread across or striped across many servers for performance or reliability reasons. The QCN algorithm is essential in data centers because it allows the ability to transfer huge amounts of data without any loss of packets When QCN is active there is almost no reduction In RTT.Congested link is fully optimized while maintaining fairness to all hosts. We are still working on the simulation of QCN to fully removed congestion control and achieving absolute fairness in Data Center network. We will also evaluate the performance of the QCN to measure the loss of packet, remove TCP incast, queue buildup, Buffer pressure for large Data Center Network. Analyzing those impairments and to prevent packet drop and ensuring the absolute fairness we try to modify the QCN algorithm and see how that works.

7.3 Discussion and Future plan

Restricting the number of TCP flows simultaneously send data to a common receiver to a reasonable value can increase TCP throughput. In this paper we propose a method to calculate the optimal value of this number. Our analysis and simulation results show that our approach can effectively utilize the link but without significant packet loss, thus eliminate the incast problem. Our approach is simple and easy to implement, and it works regardless of the width of striping and buffer size. Our approach is slightly conservative in the case of a very narrow width of striping, a very high bandwidth delay product and a very small buffer size. In the future we will extend our work for more complicated data center networks where multiple receivers on a single switch make requests across a shared subset of servers, or there are multiple Ethernet switches between receivers and servers. In realistic applications it is possible that there is some minor variance in terms of TCP senders' start times. We will address such a scenario in future work.

References

1 M. Alizadeh, A. Greenberg, D. Maltz, J. Padhye,P. Patel, B. Prabhakar, S. Sengupta, andM. Sridharan. DCTCP: Efficient Packet Transport for the Commoditized Data Center. In Proc. SIGCOMM,2010.

2 Haitao Wu, JiaboJu,Guohan Lu, ChuanxiongGuo, YongqiangXiong, Yongguang Zhang.Tuning ECN for Data Center Networks. Co-NEXT'12

3 Yan Zhang, Student MemberandNirwan Ansari. On Mitigating TCP Incast in Data Center Networks,IEEE INFOCOM 2011.

4 Quantized Congestion Notification for Multicast in DCN

5 "QCN pseudo-code version 2.0," http://www.ieee802.org/1/files/public/docs2008/au-rong-qcn-serial-hai-pseudo-code

6 HongyunZheng, ChunmingQiao. An Effective Approach to Preventing TCP Incast Throughput Collapse for Data Center Networks, IEEE Globecom 2011

7 Haitao Wu?, ZhenqianFeng?y, ChuanxiongGuo?, Yongguang Zhang?.ICTCP: Incast Congestion Control for TCPin Data Center Networks, ACM CoNEXT 2010,

8 Theophilus Benson, Ashok Anand,AdityaAkella.Understanding Data Center Traf?c Characteristics,ACM SIGCOM,2009.

9 M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable,commodity data center network architecture. In SIGCOMM,pages 63-74, 2008.

10 Y. Chen, R. Grif?th, J. Liu, R. H. Katz, and A. D. Joseph, "Understanding TCP Incast Throughput Collapse in Datacenter Networks," inProc. of the 1st ACM workshop on Research on enterprise networking,Barcelona, Spain, August 21, 2009,

11 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 46thAnnual Allerton Conference, Illinois, USA, Sep. 2008

12 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 data center communication", SIGCOMM'09, August 17-21, 2009, Barcelona, Spain

13 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 data center communication", SIGCOMM'09, August 17-21, 2009, Barcelona, Spain

14 M. Al-Fares, A. Loukissas, and A. Vahdat. A M. Alizadeh, A. Javanmard, and B. Prabhakar.Analysis of DCTCP: Stability, Convergence, andFairness. In Proc. SIGMETRICS, 2011Scalable,Commodity Data Center Network Architecture. InSIGCOMM, 2008

15 D. Abts and J. Kim. High Performance Datacenter Networks: Architectures, Algorithms, and Opportunities. Morgan and Claypool, 2011

16 M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In SIGCOMM, 2008

17 A. R. Curtis, J. C. Mogul, J. Tourrilhes, and P. Yalagandula.DevoFlow: Scaling ?ow management for high-performance networks. In SIGCOMM, 2011

18 ABU-LIBDEH, H., COSTA, P., ROWSTRON, A., O'SHEA, G.,AND DONNELLY, A. Symbiotic Routing in Future Data Centers.In ACM SIGCOMM (2010)

19 AL-FARES, M., RADHAKRISHNAN, S., RAGHAVAN, B.,HUANG, N., AND VAHDAT, A. Hedera: Dynamic Flow Scheduling for Data Center Networks. In NSDI (2010)

20 CALDERON, M., SEDANO, M., AZCORRA, A., AND ALONSA,C. Active Network Support forMulticast Applications. Network,IEEE 12

21 CASADO, M., ET AL. Ripcord: a Module Platform for DataCenter Networking. Tech. Rep. UCB/EECS-2010-93, Univeristy of California at Berkeley, 2010

22 GREENBERG, A., HAMILTON, J. R., JAIN, N., KANDULA, S.,KIM, C., LAHIRI, P., MALTZ, D., PATEL, P., AND SENGUPTA,S. VL2: a Scalable and Flexible Data Center Network. In ACM SIGCOMM (2009)

23 GUO, C., LU, G., LI, D., WU, H., ZHANG, X., SHI, Y., TIAN,C., ZHANG, Y., AND LU, S. BCube: A High Performance,Server-centric Network Architecture for Modular Data Centers.In ACM SIGCOMM (2009)

24 M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic ?ow scheduling for data center networks. In NSDI, 2010

25 J. Dean and S. Ghemawat. MapReduce: Simpli?ed data processing on large clusters. In OSDI, pages 137-150, 2004

26 R. Peterson and E. G. Sirer. Antfarm: Ef?cient content distribution with managed swarms. In NSDI, 2009

27 B. Pfaff, J. Pettit, K. Amidon, M. Casado, T. Koponen, andS. Shenker. Extending networking into the virtualization layer. In HotNets 2009

28 V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen,G. R. Ganger, G. A. Gibson, and B. Mueller. Safe and effective ?ne-grained TCP retransmissions for datacenter communication. In SIGCOMM, pages 303-314, 2009