國 立 交 通 大 學

電信工程研究所

碩 士 論 文

利用頻寬適應性技術之雲端資料中心網路壅塞控制機制

Congestion Control in Cloud Datacenter by Bandwidth Adaptation

研究生 ： 黎氏蘭香
指導教授 ： 王蒞君教授

中 華 民 國 一 百 年 八 月

國 立 交 通 大 學

電信工程研究所

碩 士 論 文

利用頻寬適應性技術之雲端資料中心網路壅塞控制機制

Congestion Control in Cloud Datacenter by Bandwidth Adaptation

研究生 ： 黎氏蘭香
指導教授 ： 王蒞君教授

中 華 民 國 一 百 年 八 月

利用頻寬適應性技術之雲端資料中心網路壅塞控制機制

Congestion Control in Cloud Datacenter by Bandwidth Adaptation

研究生：黎氏蘭香　　Student：Le Thi Lan Huong

指導教授：王蒞君　　Advisor：Li-Chun Wang

國立交通大學
電信工程研究所
碩士論文

A Thesis
Submitted to Institute of Communication Engineering
College of Electrical and Computer Engineering
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of Master of Science
In
Communication Engineering
August 2011

Hsinchu, Taiwan, Republic of China

中華民國一百年八月

利用頻寬適應性技術之雲端資料中心網路壅塞控制機制

學生：黎氏蘭香　　指導教授：王蒞君教授

國立交通大學

電機學院電信工程研究所

摘要

在過去十年來，雲端資料中心迅速崛起並成爲運算上重要的一塊，並且成爲一個重要的研究議題。然而，最近的研究者著重於解決網路設計，而網路通訊管理方面的議題仍值得深入。如何爲網路流選擇正確的路徑並不是個可以簡單回答的問題，尤其當需要保證分佈平均以及負載平衡，還要能快速的識別以及回應故障時。而更加困難的是，當這個問題還面臨著資料中心裡不確定的流量負載。

這份碩士論文意旨在研究雲端資料中心的壅塞問題。這份研究的目標在於提出一個壅塞控制機制，並評估目前的多路徑繞徑技術的效能。本文所提出的方法稱爲「頻寬適應性技術之網路壅塞控制」，該方法是基於雲端資料中心的網路流量的特性與行爲爲討論基礎。我們透過兩個模型來檢驗所提出的方法的有效性，分別是 m-port n-tree fat-tree 拓撲和英業達資料中心拓撲。模擬的情境是透過 NS-2 模擬器來實作並完成測試。其效能結果顯示，透過「頻寬適應性技術」優於雲端資料中心使用傳統 TCP 在吞吐量、封包錯誤率與延遲等問題的表現。

Congestion Control in Cloud Datacenter by Bandwidth Adaptation

A THESIS Presented to

The Academic Faculty By

Le Thi Lan Huong

In Partial Fulfillment

of the Requirements for the Degree of

Master in Communications Engineering

Institute of Communications Engineering

College of Electrical and Computer Engineering

National Chiao Tung University

August, 2011

Congestion Control in Cloud Datacenter by Bandwidth Adaptation

Student: Le Thi Lan Huong Advisor: Dr. Li-Chun Wang

Institute of Communications Engineering
National Chiao Tung University

Abstract

During the last decade, datacenters have risen to dominate the computing landscape and become an important research topic. Among four main components of a datacenter networking architecture: physical topology, routing over topology, selection between multiple paths supplied by routing, congestion control technique on the high speed, low latency network, the last three objects in the list haven't investigated worthy. In this thesis, we propose a congestion control mechanism in cloud datacenter by bandwidth adaptation and evaluates the performance of multipath routing in its present in cloud datacenter. Both fat tree and experimental topologies are considered in the testing of performance. Simulations are conducted to compare the effectiveness of the proposed mechanism. The performance results demonstrate that the proposed congestion control in cloud datacenter by dandwidth adaptation can outperform the throughput, packet error rate and delay issues compare with traditional Transmission Congestion Protocol (TCP) in cloud datacenter environments.

Acknowledgements

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis. First and foremost, my advisor, Professor Li-Chun Wang for giving me the opportunity to do the research in MC3 Lab and special thanks for his time, patience, understanding and unconditional support during the time I study here. His insights and words of encouragement have often inspired me and renewed my hopes for completing my graduate education. I would additionally like to thank Professor Hung-Pin Wen for giving me the opportunity to be part of the Inventec research.

My gratitude also goes to the Mobile Computing and Cloud Computing lab, there are not enough words to describe your excellent work and attitude in research. I learnt many things from all of you. Kapin, Weiping, Alan, Ruwan and Gorden thanks for the help in the first time I work in the cloud computing area. Dr. Sheng, thanks for your advice and for acting as a mentor to me. The most special thanks goes to my family, including my parents, my sister and brother, best partner and friend, my husband, and my little angle. You gave me your unconditional support and love through all this long process.

Contents

List of Tables

List of Figures

List of Abbreviations

ABCC	Adaptive Bandwidth Congestion Control
ACK	Acknowledgment
AD	Additive Decrease
AI	Additive Increase
AIAD	Additive Increase Additive Decrease
BIC-TCP	Binary Increase Congestion Transmission Control Protocol
CPU	Central Processing Unit
CRC	Cyclical Redundancy Check
ECMP	Equal Cost Multi Path
FTP	File Transfer Protocol
FIFO	First In First Out
GB	GigaByte (1 GB = 1,024 MegaBytes = 1,048,576 KiloBytes)
GBps	GigaBytes per second
IP	Internet Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
HSTCP	High Speed Transmission Control Protocol
NS-2	Network Simulator Version 2
NIC	Network Interface Card
OSI	Open Systems Interconnection
LAN	Local Area Network
QoS	Quality of Service
RED	Random Early Detection
RTT	Round Trip Time

STCP	Stratified Speed Transmission Control Protocol
SMSS	Sender Maximum Segment Size
TCP	Transmission Control Protocol

Chapter 1

Introduction

1.1 Thesis target and motivation

Cloud datacenters have emerged in the past few years as a new paradigm for interconnecting computing and storage in a massive scale. There are several viewpoints from which to approach the development of cloud datacenters. Network design and management are challenging problems and raise questions regarding how to choose topology, link bandwidths, and routes as well as how to quickly identify and react to failures. Many studies in cloud datacenter designs recently such as [1], [2], [3], [4] have proven the attention of network community. However, these researches only solve the problems in network design. Network traffic management aspects are still open issues. Answering questions about how to control the traffic flows and quickly identify and react to failure are not easy, these questions are more difficult due to uncertainties in traffic workloads in cloud datacenter.

The cloud datacenter offer high aggregates bandwidth and robustness by creating multiple paths in the core of the network. It's different from Ethernet in the aspect of very short distance, communications interconnection and also high bandwidth.

Due to those special points in designs and applications, traffic in cloud datacenter has very different characteristics. Researches of the traffic in cloud datacenter show a highly utilized links often. Among the 150 inter-switch links that carry the traffic of the 1500 monitored machines, 86% of the links observed congestion lasting at least 10 seconds and 15% observe congestion lasting at least 100 seconds. Short congestion periods are highly correlated across many tens of links and are due to brief spurts of high demand from the

application [5]. They lead to the truth that cloud datacenters are different from traditional enterprise network, and also cloud datacenter will experience its own congested phenomena. In such cases, congestion requires an appropriate congestion technique that can leverage the strengths of cloud datacenters.

While there are many congestion control protocols in use in the common network, the most prevalent is the Transmission Control Protocol (TCP). TCP [6] is used to provide reliable communications on top of IP and Ethernet. Since TCP employs loss-based congestion control algorithm, which continues to increase sending rate until it gets packet losses, its queuing delay can grow till its maximum and it takes time to converge to the appropriate sending rate. In very short distance communications environment as in cloud datacenters where mechanisms cannot exploit high bandwidth due to too low increasing and big decreasing in slow start and congestion avoidance phase, respectively. Recently, there are a number of TCP variants having delay-based congestion control algorithm [7], [8], [9], but they are also not very efficient because of not considering the unique congestion problem in cloud datacenter.

Due to existing protocols for wired networks like TCP and its variants are not suitable for cloud datacenter, the problem of congestion control and avoidance in multipath environment of cloud datacenter remains largely open and not clearly described yet.

This thesis aims to study the problem of congestion in cloud datacenter by utilizing unique characteristics in topology design and congestion symptoms strictly related to the philosophy and traffic property of cloud datacenter. The main objective is to examine the behavior of several cloud datacenter topologies and their impacts on congestion under traditional TCP. We will propose a new congestion control and avoidance mechanism named Adaptive Bandwidth Congestion Control (ABCC) that is different from existing traditional schemes. Some performance issues are also examined, such as throughput, end-to-end delay and packet loss rate. The simulation scenarios were implemented and tested with the use of the NS-2 simulator.

1.2 Thesis outline

In this thesis, we investigate how to design a new congestion control mechanism in cloud datacenters. We provide simulation results to show the improvements of our proposed mechanism. Chapter 2 introduces the background of cloud datacenter networking: cloud datacenter topology design, traffic management in cloud datacenter. Chapter 3 shows the system model and two case studies that will be considered to test our proposed congestion control mechanism in cloud datacenter by bandwidth adaptation mechanism, assumptions when set up the simulation environment and problem formulations. Chapter 4 analyzes the proposed scheme. Chapter 5 describes the deployment in NS2 environment and discusses the performance results in terms of throughput, error rates and delay comparisons between single path, equal cost multi path (ECMP) and adaptive bandwidth congestion control (ABCC). Chapter 6 gives some concluding remarks and future work.

Chapter 2

Background

2.1 An overview on cloud datacenter networking

In recent years, cloud computing has become more popular. As a result, cloud datacenters have risen to dominate the computing landscape. Today's largest datacenter have hundreds of thousands servers, and run distributed applications that spread computation and storage across many thousands of machines. With so many hosts, it is impractical to manually manage the allocation of task to machines. While applications may be written to take advantage of locality within the datacenter, large distributed computations inevitably are spread across many racks of machines. As a result, the network can often be the bottleneck. Next part will review the background for networking issues and techniques in cloud datacenters, including topology, addressing, routing, forwarding loops, and multiple paths.

2.2 Cloud datacenter network design

2.2.1 Topology

A typical architecture for connecting hosts in a cloud datacenter is three tiers architecture, consisting of core, aggregation, and access tiers. Core and aggregation tiers require higher bandwidth than the access tier. The design of the three tier architecture needs to consider the so called oversubscription ratio issue, defined as the sum of the bandwidth required from all the end hosts in the access level over the provided bandwidth in the upper level [10].

The higher value of oversubscription ratio leads to higher blocking and lower throughput. In a large-scale datacenter, the typical oversubscription ratios are 2.5 up to 8. To reduce the oversubscription ratios, high-bandwidth equipments will be required in core and aggregation levels. However, the high-bandwidth equipments are high cost. Fat-tree topology [11] provided low oversubscription ratio using low-cost equipment.

2.2.2 Addressing

Addressing in inter-connected network can be classified into layer-2 and layer-3 addresses. MAC addresses are defined in the layer-2 network, while IP addresses are defined in layer-3. The structure of layer-2 MAC addresses is flat, which is different from the hierarchical structure of layer-3 IP addresses. Usually, the layer-3 IP addresses are easier for management and scalability.

2.2.3 Cost and equipment

The networking equipments employed in a cloud datacenter spend 15% of the total cost [12]. The price difference between the low-end and high-end equipment is huge, thereby providing a strong incentive to build large-scale data center networks from low-price networking equipments. For example, a 48-port GigE switch costs $7,000, and a 128-port 10 GigE switch costs $700,000 [10]. Adopting commodity networking equipments also considered in the switching design fifty year ago. Charles Clos designed a network topology that delivers high levels of bandwidth for many end devices by appropriately interconnecting cheaper commodity switches [13]. A special instance of a Clos topology called the fat-tree topology [11], which was shown to reduce four to five times cost of networking equipments [10][14] . Beside the commodity devices, cloud datacenter networking researches [6][7] also adopt NetFPGA [15] as an experimental networking equipment (e.g. modified Ethernet switches and IP routers). NetFPGA is a programmable PCI card which can process packets at line-rate on four 1Gbps-ports. Researchers can create their designs in Verilog code, and then download them to the NetFPGA board. The prices of NetFPGA boards are available at $500 [15]. That price is acceptable.

2.2.4 Routing mechanism

Multipath routing is the desirable property in the network design, which can provide fault tolerance, load balance, and bandwidth enhancement. Equal cost multipath protocol (ECMP) [20] is a widely used in the current networks. Nevertheless, the issues of packet reordering and unfair flow splitting in ECMP are remained to be solved if it is implemented in cloud datacenters.

2.3 Traffic management in datacenter

Network design and management are challenging problems and raise questions regarding how to choose topology, link bandwidths, and routes as well as how to quickly identify and react to failures. Many studies in datacenter network designs recently such as [1], [2], [3], [4] have proven the attention of network community. However, these researches only solve the problems in network design and the network traffic management aspects are still open issues. Answering questions about how to choose right path for flows to guarantee fair distribution and load balancing and can quickly identify and react to failure are not easy. These questions are further made difficult due to uncertainties in traffic workloads in datacenters. Some studies continue the management aspect in cloud datacenter by providing the solutions for load balancing with multipath, congestion control, combining multipath routing and congestion control. We will consider each part individually in the following sections.

2.3.1 Multipath routing for load balancing

Load-sharing is an important technique in communication networks. It allows a network device (router, switch) to distribute the outgoing and incoming traffic among multiple paths. Load-sharing can be achieved in two ways: through congestion-aware routing algorithms to route the specific demand or routing the packets of the same demand over multiple paths along the way. The latter called multipath routing provides fast resiliency as well as finer degree of load sharing in the network. Current routing schemes typically focus on discovering a single optimal path for routing, according to some desired metrics. Accordingly, traffic

is always routed over a single path, which often results in substantial waste of network resources. Multipath routing is an alternative approach that distributes the traffic among several good paths instead of routing all traffic along a single best path. Multipath routing represents a promising routing method to achieve load balancing and is more resilient to route failures. Multipath is not a new idea which is popular in wireless networks. Many multipath routing protocols such as SMR, AODV, AOMDV, and AODV multipath have been proposed and performance evaluations of these protocols showed that they achieve lower routing overhead, lower end-to-end delay and alleviate congestion in comparison with single path routing protocols. In wireless networks, multipath routing protocols establish multiple disjoint paths from a source to a destination, thereby improving resilience to network failures and allowing for network load balancing. These effects are particularly interesting in networks with high node density (and the corresponding larger choice of disjoint paths) and high network load (due to the ability to load balance the traffic around congested networks). Fortunately, all above properties are right matched with the design of datacenter network, multipath routing attracts the attention of network management designer by state of art ECMP [16] and VLB [17]. Portland [2] adopts ECMP why VL2 [4] combines two related mechanisms: VLB and ECMP for random traffic spreading over multiple paths. Anyway ECMP is per flow static hashing. It can cause substantial bandwidth losses due to long term collision. On the other hand, this static mapping flow-to-path does not account for either current network utilization or flow size. With resulting collisions overwhelming switch buffer and regarding overall switch utilization. HFMF [18] used adaptive flow-splitting schemes according to the corresponding level in the three-tier fat-tree topology. Comparison results proved that HFMF can reduce the packet disorder arrival and achieves a better performance of load balancing compare with ECMP. By the way, multipath in cloud datacenters requires more research to leverage the diversity of connections in its design.

2.3.2 Congestion control

Congestion typically occurs where multiple links feed into a single link where internal LANs are connected to WAN links. Congestion also occurs at routers or switches in core networks where nodes are subjected to more traffic than they are designed to handle. In cloud datacenter, congestion happens at edge switch where is the point of a connection of dozens of servers

and core routers. Researches of the traffic in cloud datacenter show a highly utilized links happen often. Among the 150 inter-switch links that carry the traffic of the 1500 monitored machines, 86% of the links observe congestion lasting at least 10 seconds and 15% observe congestion lasting at least 100 seconds. Short congestion periods are highly correlated across many tens of links and are due to brief spurts of high demand from the application [5]. It clearly shows that cloud datacenters are different from traditional enterprise network, and also cloud datacenter will explicit its own phenomena when congested.

There are two commonly type of traffic in network: UDP and TCP. While UDP is typically used for real-time audio and video streams, because there is no need to recover lost packets. UDP is an unreliable transport protocol that does not send ACK signals back to the source, TCP is a connection-oriented protocol. Thus congestion control adopts only on TCP. TCP traffic is of particular importance in any networks as it currently carries the great majority (more than 90%) of network traffic. We will give an overview of congestion control mechanisms in the TCP protocol next part.

2.3.2.1 An Overview of Congestion Avoidance in TCP

TCP is one of two protocol standards commonly referred to as TCP/IP. TCP sits on top of the IP layer and delivers segments onto the IP layer for further processing. These segments are then delivered onto the lower level layers and finally on to the network. TCP was officially adopted as a standard in RFC 1793 in 1981 and was designed to deal with flow control and error correction traffic in network, TCP can ensure reliable delivery of a message from a source to a destination application.

To understand how TCP works, the general features of TCP will be clearly described. In the following part, we also provide an overview of features a TCP variant can possess. These features will be dealt with in historical and chronological manner. In order to describe the TCP protocol, some concepts are needed:

* Round Trip Time (RTT): Assume that there is no packet loss in network, RTT is the required time for a packet to be sent from a source to a destination application and for the corresponding acknowledgement to be received by the source appk.

* Advertised window (awnd): awnd is the value of the amount of data that a destination has announced that it is willing to receive. This is advertised to source in every ACK sent

by the destination.

* Congestion window (cwnd): Assuming that the source is not restricted by the advertised window, in that case, cwnd represents for the number of packets transmitted but not yet acknowledged.

* Send window (swnd): sometimes denoted as w, it is the minimum of awnd and cwnd.

* Additive Increase Multiplicative Decrease (AIMD): this is a name for the technique that TCP increases its congestion window by adding to it on receipt of ACK's and decreases the send window by a multiplicative factor on receipt of an indication of packet loss.

* Slow Start: In Slow Start mode the congestion window is doubled whenever it receives ACK packet of previous sending packet. This mechanism creates an exponential shape of increasing rate.

* Slow start threshold (ssthresh): state variable is used to determine whether the slow-start or congestion avoidance phase is used to control data transmission. If $cwnd <= ssthresh$ the sender will follow the slow start phase. If $cwnd > ssthresh$, it will switch to congestion avoidance phase.

* Fast Retransmit: a mechanism whereby the source retransmits a packet right after receiving a number of duplicate ACK's rather than waiting for a retransmit timer to timeout. By that way, it can reduce the time that a sender waits before retransmitting a lost segment.

* Fast Recovery: the mode entered after a packet drop is detected via Fast Retransmit. In this mode when packets (recognized through 3 duplicate ACKs) are not received, the congestion window size is reduced to the slow-start threshold, rather than the smaller initial value.

* Retransmission Time-Out (RTO): it is calculated based on current RTT and RTT variance. It's represented for an interval time that a source has to wait without receiving an ACK before marking a packet as lost, retransmitting it and entering Slow Start mode.

* Congestion Avoidance: the mode the source enters after Fast Recovery. The source uses an AIMD strategy, linearly increasing its congestion window at a rate of one packet per RTT.

* Maximum Sized Segment (MSS): the maximum packet size that can be sent by a TCP source. During the initial phase of a TCP connection the receiver (or receivers when the data flow is bi-directional) provides details of the amount of incoming data that it can process.

This informs the source of the maximum data that the destination is currently willing to receive.

Originally TCP's flow control was adjusted by (1st) the maximum allowed window size advertised by the receiver and (2nd) the policy that allowed the sender to send new packets only after receiving the acknowledgement for the previous packet. To overcome the disadvantages due to the sluggishness of TCP, Tahoe TCP was released including three congestion control algorithms: slow start, congestion avoidance and fast retransmit. After that, Reno TCP provides one more algorithm called fast recovery. It introduces receiver's advertised window, awnd, which is used to prevent the sender from overrunning the resources of the receiver. TCP's congestion control also includes two new variables for the connection. The congestion window, cwnd, is to prevent the sender from sending more data than the network can accommodate in the current load conditions. The slowstart threshold, ssthresh, is the estimate for the available bandwidth in the network. The window size of the sender, w, was not only defined by the maximum allowed window size advertised from sender, but also set by another parameter: congestion window: w = min (cwnd, awnd).

The main idea of TCP congestion control is to modify cwnd adaptively to reflect the current load of the network. In practice, congestion phenomenon is detected by the lost packets. A packet loss can basically be detected either via a time-out mechanism or via duplicate ACKs.

* Timeouts: Associated with each packet is a timer. If the time expires, timeout occurs, and the packet is retransmitted. The value of the timer, denoted by RTO, should ideally be of the order of an RTT. However, as the value of RTT is not known in practice, it is measured by the TCP connection by using, e.g, the so called Jacobson/Karels algorithm.

* Duplicate ACKs: If a packet has been lost, the receiver keeps sending acknowledgements but does not modify the sequence number field in the ACK packets. When the sender observes several ACKs acknowledging the same packet, it concludes that a packet has been lost.

The following parts describe basic phases in TCP: Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery. These phases will be included in the next sections to describe TCP Variants.

* Slow Start and Congestion Avoidance: In slow start phase, when a connection is

established, the value of cwnd is first set to 1 and after each received ACK the value is updated to $cwnd = cwnd + 1$ (exponential growth). The growth of cwnd continues until a packet loss is observed. Whenever a packet loss is detected, the value of ssthresh is updated to $ssthresh = cwnd/2$. After the packet loss, the connection starts from slow start again with $cwnd = 1$, and the window is increased exponentially until it equals ssthresh. At ssthresh point, the connection goes to congestion avoidance phase where the value of cwnd is increased less aggressively with the pattern $cwnd = cwnd + 1/cwnd$ (linear growth). This linear increasing will continue until a packet loss is detected.

* Fast Retransmit: Duplicate ACKs that were mentioned to be one way of detecting lost packets can also be caused by reordered packets. When receiving one duplicate ACK the sender cannot yet know whether the packet has been lost or just gotten out of order but after receiving several duplicate ACKs it is reasonable to assume that a packet loss has occurred. The purpose of fast retransmit mechanism is to speed up the retransmission process by allowing the sender to retransmit a packet as soon as it has enough evidence that a packet has been lost. This means that instead of waiting for their transmit timer to expire, the sender can retransmit a packet immediately after receiving three duplicate ACKs.

* Fast Recovery: In Tahoe TCP the connection always goes to slow start after a packet loss. However, if the window size is large and packet losses are rare, it would be better for the connection to continue from the congestion avoidance phase, since it will take a while to increase the window size from 1 to ssthresh. The purpose of the fast recovery algorithm in Reno TCP is to achieve this behavior. In a connection with fast retransmit, the source can use the flow of duplicate ACKs to clock the transmission of packets. When a possibly lost packet is retransmitted, the values of ssthresh and cwnd will be set to $ssthresh = cwnd/2$ and $cwnd = ssthresh$ meaning that the connection will continue from the congestion avoidance phase and increases its window size linearly.

2.3.2.2 TCP variants

In order to understand the current status of TCP it is important to look at its development and in particular the reasoning behind specific design features. Early deployments of TCP used a go-back-N model (sending sequence goes back N packets) when packets were lost. These implementations had no congestion control and led to a series of serious congestion

problem on the Internet for long time. During these congestion collapses the data throughput of connections was severely reduced due to excessive retransmission of packets. These issues were addressed by a version of TCP called Tahoe [23] in which the problem of congestion was approached by a "Conservation of Packets" principle whereby new packets were not put into the network until the old ones left. TCP Tahoe was the first method to employ three mentioned transmission phases: slow start, congestion avoidance, and fast retransmit and is thus a self clocking system, formed by the transmission of data and the receipt of acknowledgements.

Even thought Slow Start and Congestion Avoidance are implemented together, they are independent algorithms with different objectives. Slow Start probes the network so that the TCP source can get an initial indication of the network bandwidth available. In practice, each TCP variants implement this step in different ways. Congestion Avoidance more gently probes the network so that the TCP source can adapt to change network conditions. A TCP connection will start in Slow Start mode but switch to Congestion Avoidance mode after cwnd reaches the value of ssthresh. In addition to these enhancements Tahoe also includes Fast Retransmit, better RTT variance estimation, and exponential retransmit timer back-off. These enhancements dramatically enhanced the throughput performance of TCP [24]. In practice, each TCP variants implement these steps in different ways.

Reno TCP [25] is the next major variant of TCP. This is similar to the Tahoe TCP, except it also includes Fast Recovery. Reno TCP does not return to Slow Start after Fast Recovery (which ends on the receipt of the retransmitted packet). It reduces the congestion window to reduce the current window size to one half. In this example the TCP flow goes into Timeout mode following Slow Start due to excessive packet transmission during Slow Start phase. The special point in Reno is delayed ACKs including. The two above mechanisms experience poor performance when multiple packets are lost from one window (cwnd) of data. With less information available from cumulative acknowledgments, TCP Tahoe and Reno sources can only learn about a single lost packet per round trip time. An aggressive source could choose to retransmit packets early, but such retransmitted packets may have already been successfully received. To address this issue, TCP NewReno [25] modifies the action taken when receiving new ACK's. In order to exit Fast Recovery, the TCP NewReno source must receive an ACK for the highest sequence number sent before entering Fast

Recovery. Thus, unlike TCP Reno, new partial ACK's do not take TCP NewReno out of Fast Recovery phase. In this way, Reno retransmits one packet per RTT until all lost packets are retransmitted.

TCP NewReno almost addresses the problem of multiple dropped packets but it does not use all the information on loss information available at the receiver. This issue is addressed by the Selective Acknowledgment (SACK) mechanism [26], combined with a so called selective repeat retransmission policy. The receiving TCP sends back ACK packets to the source. This can inform the source of data all necessary information about dropped packets. The source can then retransmit packets that have been dropped. This scheme has the benefits of allowing the source to intelligently retransmit packets and react to multiple dropped packets efficiently.

2.3.2.3 TCP variants for high speed networks

In particular, recent research has identified the following challenge faced by the conventional transport control protocol: TCP cannot satisfy large data transfer requirements for future data networks because it is unable to efficiently utilize the huge bandwidth provided by high speed.

To understand the above identify let us consider Figure 2.1, which shows the bandwidth utilization of links with several TCP connections, each one is 1KB in packet size. The 5 link capacities are: 155Mbps, 622Mbps, 2.5Gbps, 5Gbps, 10Gbps. It is set up in NS2 simulation with DropTail queuing model, the queue is set at 100 packets. Figure 2.1 clearly shows a big descreasing of link utilize when link capacity increases. The result proves that TCP cannot fully utilize the huge capacity of high speed network.

There are several TCP variants for addressing these issues. We consider here three of TCP variants: AIMD, STCP and HSTCP.

* The additive increase/multiplicative-decrease (AIMD) algorithm is a feedback control algorithm used in TCP Congestion Avoidance. AIMD combines linear growth of the congestion window with an exponential reduction when congestion takes place. The taken approach is to increase the transmission rate (window size), probing for usable bandwidth, until loss occurs. The policy of additive increase may, for instance, increase the congestion window by 1 MSS (Maximum Segment Size) every RTT (Round Trip Time) until a loss packet is de-

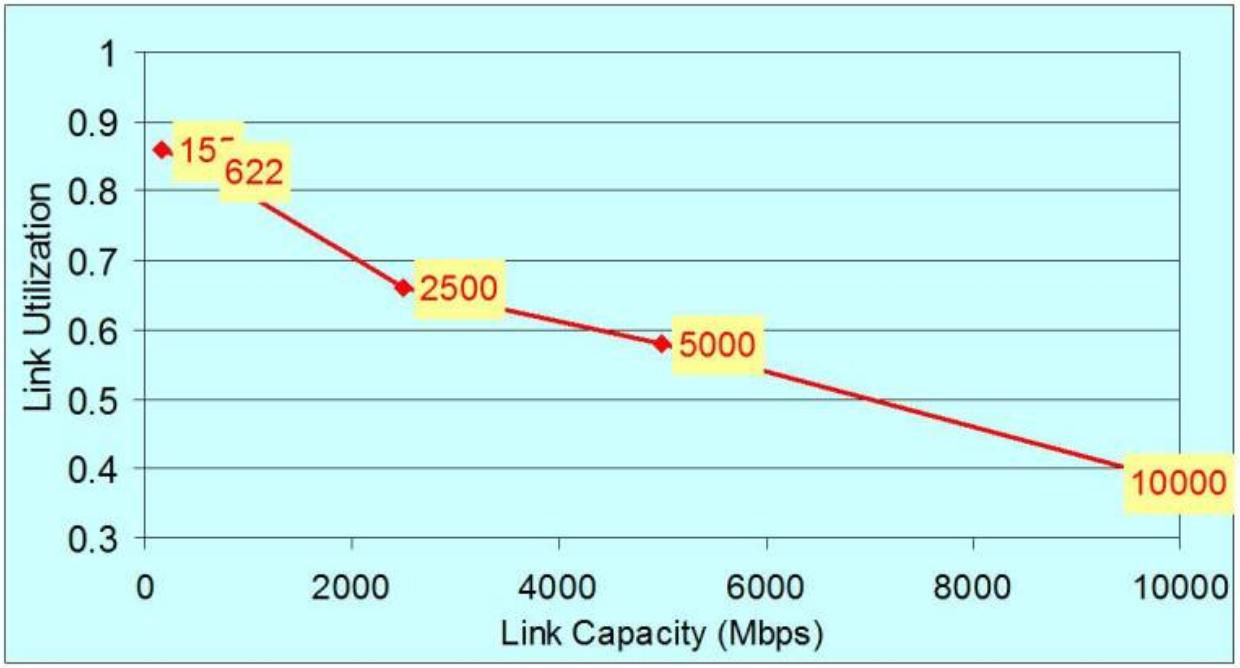


Figure 2.1: Bandwidth utilization by TCP in different link capacities.

tected. When the lost is detected, the policy is changed to be one of multiplicative decrease, which may, for instance, cut the congestion window in half after loss.

* Stratified TCP (STCP)[28] is a protocol designed to utilize bandwidth in an aggressive and efficient way. It is based on the principle of virtual layers. In this technique, data transmission starts off with a single layer (as in conventional TCP) and the layers are gradually increased based on certain criteria. To determine the criteria for increasing the number of layers, STCP applies the TCP sending rate equation and modifies the equation to make it more dynamic to the changing network conditions. Hence, STCP owns following major features: aggressive bandwidth utilization and responds to network changes.

* HighSpeed TCP [27] is a modification to TCP congestion control mechanism, proposed by Sally Floyd from ICIR (The ICSI Center for Internet Research). The main problem of TCP connection is that it takes a long time to make a full recovery from packet loss for high-bandwidth long-distance connections. Like the others, new TCP implementations propose a modification of congestion control mechanism for use with TCP connections with large congestion windows. For the current standard TCP with a steady-state packet loss rate p, an average congestion window is $1.2/\sqrt{p}$ segments. It places a serious constraint in realistic environments.

We summarize some TCP variants and provide comparisons of TCP and our work in the table 2.2.

2.4 Summary

In this chapter we provided an overview on cloud datacenter, especially topology design and traffic management. We described multipath routing technique in cloud datacenter and also gave a briefly summary of congestion voidance control in network. We note that there are many variants of TCP. The main variants of TCP in use today are favours of TCP Reno and TCP Sack. For high speed networks, many other TCP variants proposed, such as AIMD, STCP, HSTCP. The main difference between these TCP variants lies in the manner in which they deal with lost packet recovery. However, there's no TCP variant pays attention to the congestion problem in cloud datacenters. Hence, there is a need to study how to enhance the TCP performance in high speed and low latency cloud datacenter networks. In this work, we propose a new TCP protocol, namely Adaptive Bandwidth Congestion Control (ABCC) which modifies the TCP behavior with more aggressive bandwidth probing and better utilization. A brief introduction to ABCC is given in the chapter 4.

Table 2.1 will explain what we do in the rest of thesis.

Table 2.1: Literature Survey

	Topology	Multipath	TCP variants	Simulation
VL2	Fat tree	o	x	o
PortLand	Clos	o	x	o
Experimental	Multi-root	x	x	x
Our work	**Fat tree, Inventec**	**o**	**o**	**o**

Table 2.2: Comparisions of TCP Variants and our proposed

Types	Slow Start Phase	CA without Loss	CA with Loss
TCP	$cwnd = cwnd + 1$	$cwnd = cwnd + \frac{1}{cwnd}$	$cwnd = cwnd\left(1 - \frac{1}{2}\right)$
AIMD	$cwnd = cwnd + 32$	$cwnd = cwnd + \frac{1}{cwnd}$	$cwnd = cwnd\left(1 - \frac{1}{2}\right)$
STCP	$cwnd = cwnd + cwnd \times 0.01$	$cwnd = cwnd + \frac{1}{cwnd}$	$cwnd = cwnd\left(1 - \frac{1}{8}\right)$
HSTCP	$cwnd = cwnd + inc(cwnd)$	$cwnd = cwnd + \frac{1}{cwnd}$	$cwnd = cwnd + dec(cwnd)$
Our work	**Bandwidth Adaptation**	**Update Sending Rate**	$\boldsymbol{cwnd = cwnd\left(1 - \frac{1}{\delta}\right)}$

Chapter 3

System Model and Problem Formulation

3.1 System Model

3.1.1 Model

The network model under consideration in this research is multi-root tree network topology in cloud datacenters. We focus on the simulation of proposed mechanism in two models: a m-port n-tree fat-tree topology and an experimental datacenter. The two models belong to typical cloud datacenter architectures that are using widely in many datacenters nowadays. They commonly consist of three-level trees of switches or routers. A three-tiered design has a core tier in the root of the tree, an aggregation tier in the middle and an edge tier at the third level of the tree. In the edge level or aggregation level, every device has two outlets for the connection of high level devices.

3.1.1.1 Model 1: Fattree topology

There are many ways to construct the fat-tree network topology. The root switches in traditional k-ary n-trees only use half of their communication ports. In order to utilize every port of root switches, we use an m-port n-tree approach [22] to construct the cloud datacenter. In this construction, the number of ports in each switch is denoted by m, and n is signed for the number of tiers in network. An m-port n-tree consists of $2 \times (m/2)^n$ hosts

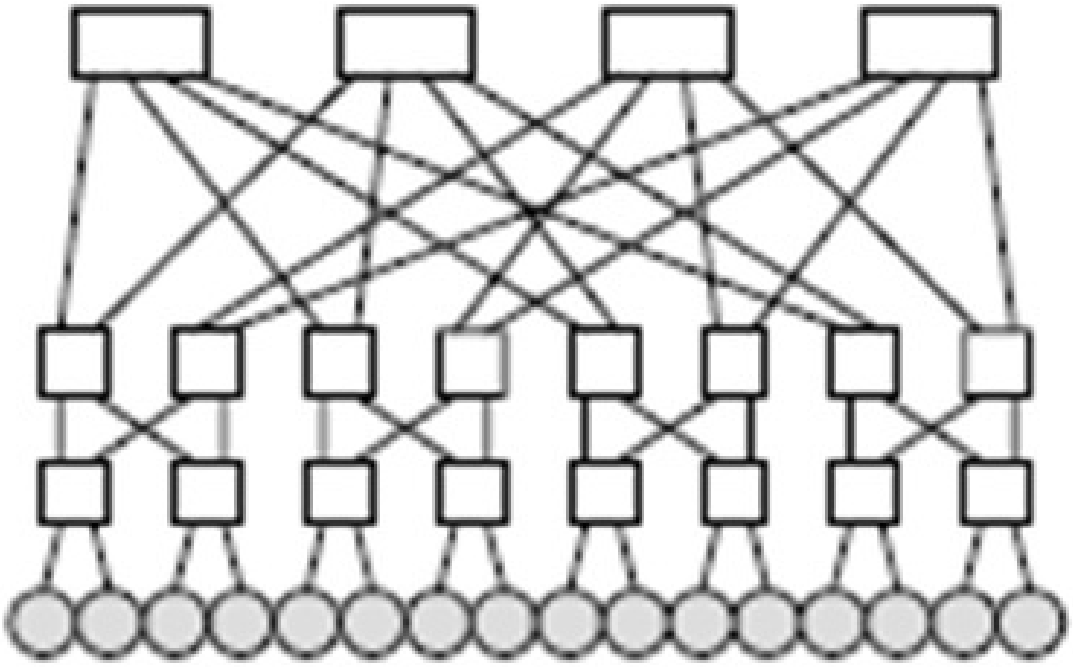


Figure 3.1: 4-port 3-tree Fat tree topology with 16 hosts and 20 switches.

and $(m/2)^{n-1} + m^{n-1}$ switches. Each switch has m communication ports 1, 2, 3,..., m that are attached to other switches or hosts. There are $(m/2)^{n-1}$ root switches. Every switch other than root switches uses port 1, 2,..., $m/2$ connecting with its descendants or hosts, and uses port $m/2$+1, $m/2$+2,..., m connecting with its ancestors. Hosts are connected at leaf switches. In this study, we evaluate the proposed mechanism in two m-port n-tree fat-tree topologies, a 4-port 3-tree constructed for small scale and 8-port 3-tree one for medium scale cloud datacenter. The large-scale cloud datacenter is considered in the experimental topology. Small-scale datacenter consists of 20 1Gb switches with 4 ports, where 4, 8 and 8 switches are in core, aggregate and edge tiers, respectively. This network serves for 16 hosts.

Medium-scale datacenter consists of 80 8-ports 1Gb switches with 16 in core, 32 in aggregate and 32 in edge tier, respectively. This network serves for 128 hosts.

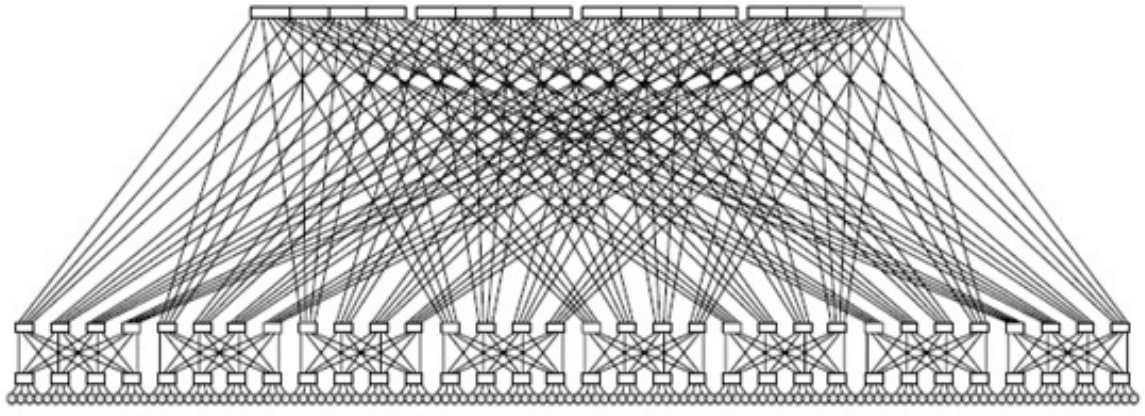


Figure 3.2: 8-port 3-tree Fat tree topology with 128 hosts and 80 switches.

3.1.1.2 Model 2: Experimental topology

A real datacenter designed by Inventec Inc. has 65 switches. There are 5 switches (10G, 48 ports) in core tier, each switch connects to 10 switches in Aggregate (Region) tier and remains 10 links to 10 switches in region tier. Under the core tier, the network is divided into two regions. Each region has 5 switches (10G, 48 ports) which aggregates 5 racks consist of 5 switches and 1 data node. Four switches in aggregate tier remain fully down-links to each rack; there are always 2 links from 2 switches in aggregate tier to data node in each rack. Each switch also connects to 10 data nodes. Each switch in region tier always has 5 up-links to connect to core tier.

3.1.2 Assumptions

The multi-root cloud datacenter network is a packet-switching based network. Routing in a subnet is based on the forwarding table stored in each switch. When a packet arrives in a switch, the packet will be forwarded to the corresponding output port via the forwarding table lookup. The table look up will find the destination in the shortest path condition or send same content in the shortest path and one or several backup paths. When multiple hosts want to send packets to the multiple other hosts at the same time, the link congestion may occur while the bandwidth of multiple paths offered by the multi-root topology is wasted. We assume applications that generate a series of relatively short data transfers for that reason, agile rate adaptation is very important to improve communication efficiency.

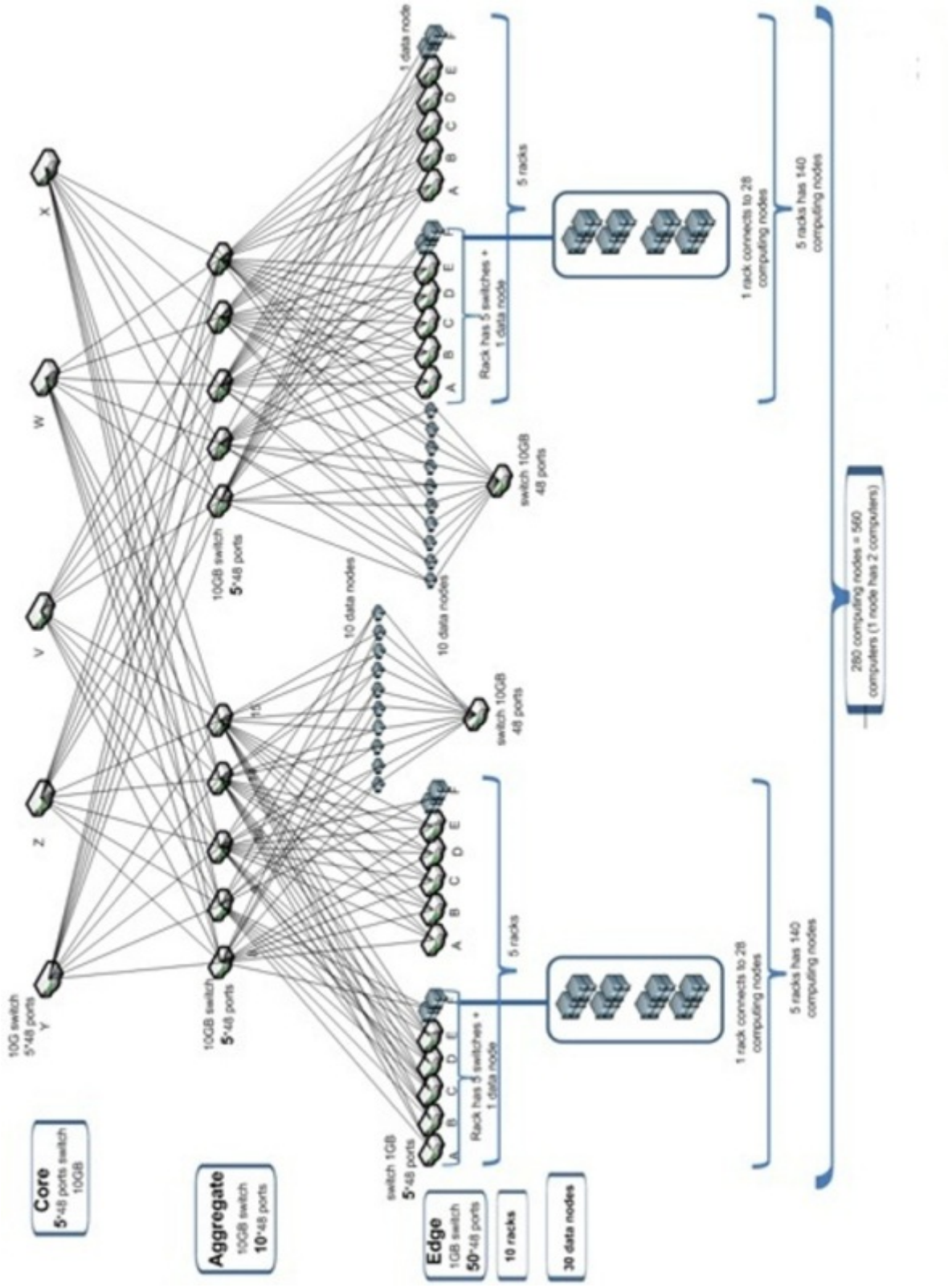


Figure 3.3: An experimental cloud datacenter with 65 switches and 1000 hosts.

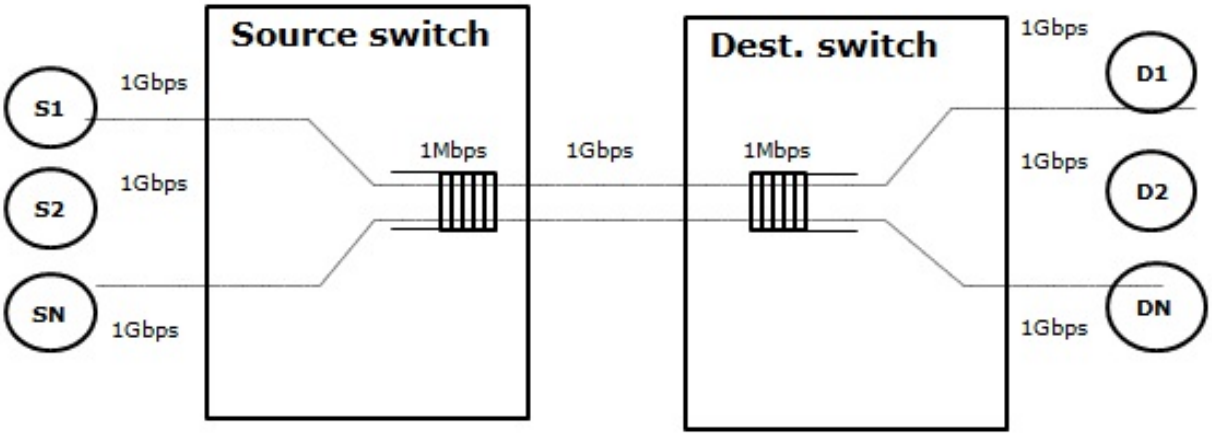


Figure 3.4: Assumption model.

Traffic will be created to fully load the network under all-to-all traffic with large and small distributions so that we can observe the congestion phenomenon.

3.2 Problem Formulation

3.2.1 Goal

The goal of our research is to propose a multipath routing combined with congestion control in datacenters which can improve throughput, acceptable delay and minimize packet lost in cloud datacenter. Cloud datacenters are different from traditional enterprise networks. In addition, cloud datacenters will face its own congested phenomena. Existing protocols for wired networks like TCP and its variants are not suitable for cloud datacenters. The problem of congestion (control and avoidance) in cloud datacenter remains largely open and not clearly described yet. This thesis aims to study the problem of congestion and consider the performance strictly related with the multipath environment in the design of cloud datacenters. The main objective is to design a new congestion control mechanism based on the behavior and characteristics of traffic in cloud datacenter and then put the proposed mechanism in the multipath routing network environment. It aims to evaluate the performances: throughput, packet lost rate and delay in different cloud datacenter topologies in the comparison with existing traditional scheme. The simulation scenarios were implemented and tested by using the NS-2 simulator.

3.2.2 Overview of multipath routing and congestion control in cloud datacenter

We consider the network with several hosts connecting to source switches. These computers would like to send the traffic through a multi-path network environment in which all paths have the same cost. We do the communication by transferring through an Equal Cost Multi Path (ECMP) routing. For the purpose of congestion control in the network, we design an Adaptive Bandwidth Congestion Control (ABCC) mechanism and implement in the presence of Multi Path routing environment.

ABCC is a protocol designed to utilize bandwidth in an adaptive and efficient way. It is based on the principle of real time sending rate updating. In this technique, data transmission starts in slow start phase as in conventional TCP and sending rate are quickly increased based on specific calculating. To determine the criteria for increasing or decreasing the rate of traffic, ABCC applies the updating sending rate equation and modifies the equation when congestion happens. The purpose is to make it more dynamic when network conditions change. Hence, ABCC has four major features:

* Quickly achieve bandwidth utilization,
* Respond to network changes by update sending rate,
* More flexible with error detection in network,
* Keep the basic principle of TCP congestion control unchanged.

Details of the mechanism will be introduced clearly in the next chapter.

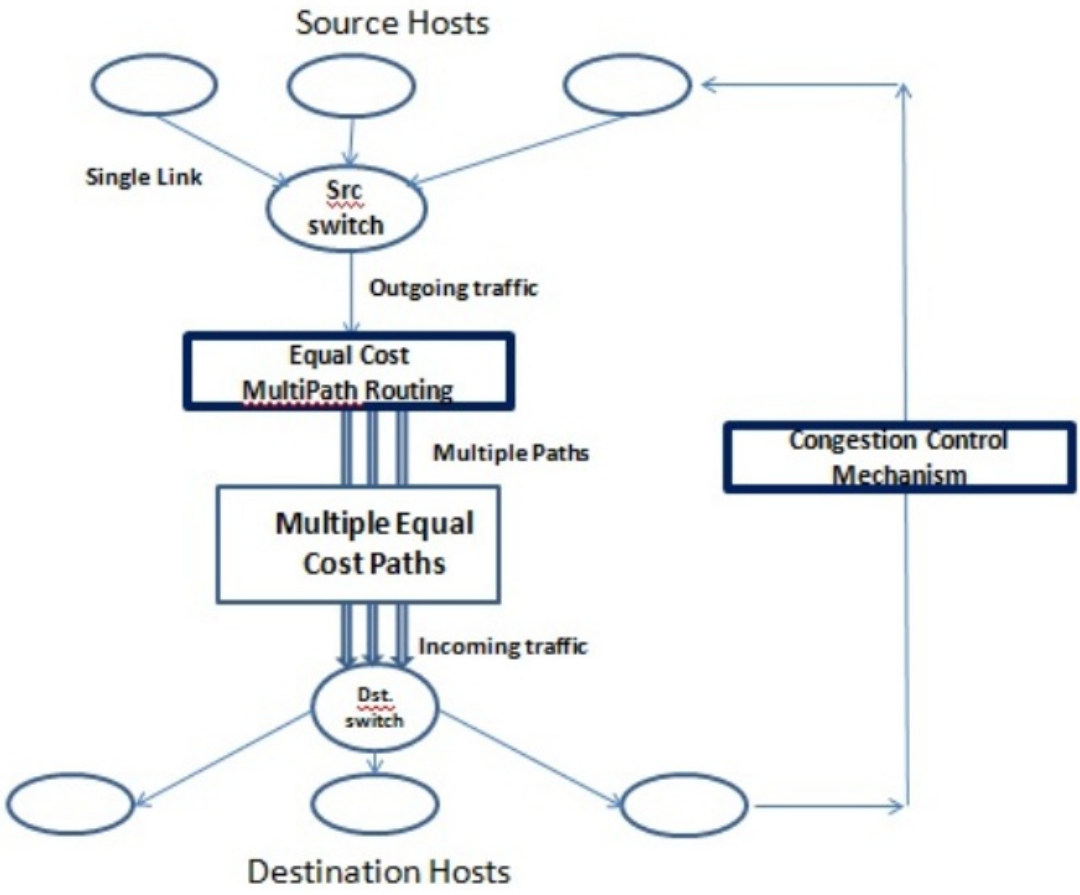


Figure 3.5: Congestion control in multipath datacenter.

Chapter 4

Adaptive Bandwidth Congestion Control Mechanism

Fig. 4.1 shows an overview of the proposed mechanism. We assume that an upper-layer application sends a group of packets to the proposed mechanism, which is located at the transport layer. The application will get the highest quality of transmission if it can be transferred at the highest capacity in the network layer. The proposed mechanism tries to achieve this demand. The proposed congestion control algorithm starts whenever a connection starts transferring or a connection time out. The algorithm modifies three states corresponding with slow start phase, no packet loss and packet loss detection in congestion avoidance phase in TCP.

The algorithm is started by the sender whenever it detects a new started connection. When a new group of packets arrives at the sender host, it will check the value of sending rate in R. If $R = 0$ then the algorithm will switch to Rate Prediction State to initial establish the sending rate.

4.1 Rate Prediction State

When a new connection starts, as shown by a new group of packets arrives, the algorithm will do the two following tasks in parallel.

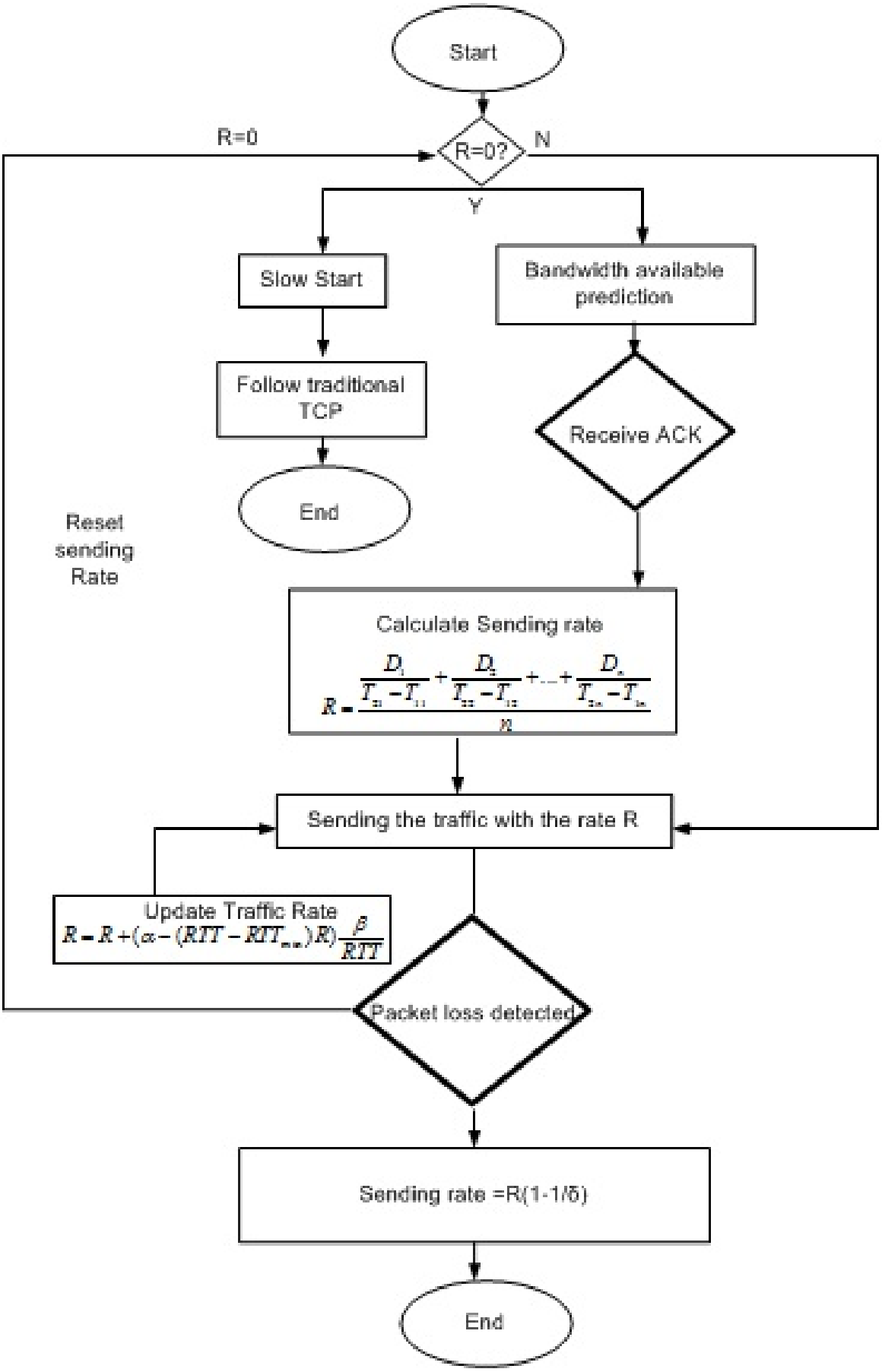


Figure 4.1: Adaptive Bandwidth Congestion Control Mechanism.

4.1.1 Initial evaluate sending rate

First n groups of packets are sent out after a random interval individually that don't need to wait for their previous ACKs. Each group has N packets that randomly chosen in range (x,y). This is the number of packets that a sender can transmit in the cloud datacenters under the limitation of discarding when an interface runs out of resources. In our proposed mechanism, the resource is the memory for queuing . After receiving all ACK packets, the sender host calculates the instantaneous sending rate by the following formula:

$$R = \frac{D}{T_2 - T_1} \tag{4.1}$$

We have n groups then the sending rate can be written by:

$$R = \frac{\frac{D_1}{T_{21}-T_{11}} + \frac{D_2}{T_{22}-T_{12}} + ... + \frac{D_n}{T_{2n}-T_{1n}}}{n} \tag{4.2}$$

where:

* R: sending rate

* D_1, D_2,..., D_n : amount of data sent in the group 1,2,...,n

* T_1 ,T_2 : reception time of the first and last ACK packets, respectively in each group.

As a result, $T_2 - T_1$ is regarded as the time for the data packets to traverse through the entire path from sending host to receiving host (or sessions). Thus, (4.2) estimates the average sending rate and reflects the instantaneous available bandwidth of the network at that time. Although the calculated initial sending rate would be useful if only use one first group of packets, it should be more exactly to verify by sending n groups to get the exact sending rate.

4.1.2 The rest traffic

Following the Slow Start Phase in TCP, the algorithm begins in the exponential growth phase initially with a congestion window size of 1 or 2 segments, which is increased by one segment size for each ACK received. This state is done during the time the sender waiting for the result from the above calculation to get exactly the sending rate that can be achieved. During this time, the sender seems to blind the residual bandwidth. For the reason of on-time transmission, data should be sent without any delay. Right after the sending host

(session) get the value of sending rate R, the sender now forces all the data flows following the transmitting as the sending rate R. By this method, sender can rapidly increase sending rate when the network path is under-utilized.

4.2 Updating Sending Rate

The purpose of this step is to achieve the target throughput in real time by update sending rate. In high speed and short distance datacenter, TCP requires a very high sending rate to efficiently utilize the network resource. However, the standard TCP increases its congestion window by one packet for each ACK received and reduces it by half on packet loss detection. Because the bandwidth delay product is large in cloud datacenters, it requires a really long time for TCP to achieve transmitting at the maximum capacity of the network. We almost solve the above problem by the mechanism of specifying the instantaneously sending rate of traffic. How to keep updating the bandwidth available is also a big challenge. The updating sending rate should have the following properties: (1) it should have an aggressive, scalable increase rule when the network is sensed to be under-utilized; and (2) it should also reduce sending rate accordingly when the network is sensed to be fully utilized. Our updating of sending rate is derived from TCP Vegas. A state variable, called baseRTT, is maintained as an estimation of the transmission delay of a packet over the network path. When the connection is started, baseRTT is updated by the minimal RTT that has been observed so far. An exponentially smoothed RTT, sRTT, is also maintained. Then, the number of backlogged packets of the connection can be estimated by following algorithm:

$$Expected = \frac{w}{baseRTT}$$

$$Actual = \frac{w}{RTT}$$

$$Diff = baseRTT * (Expected - Actual)$$

where:

w: window size of the sender.

The *Expected* gives the estimation of throughput we get if we do not overrun the network path. The *Actual* stands for the throughput we really get. Then, $(Expected - Actual)$ is

the difference between the expected throughput and the actual throughput. Multiplying by baseRTT, it stands for the amount of data that injected into the network in last round but does not pass through the network in this round, i.e. the amount of data backlogged in the bottleneck queue. An early congestion is detected if the number of packets in the queue is larger than a threshold γ. If $Diff < \gamma$, the network path is determined as under-utilized; otherwise, the network path is considered as busy and delay-based component should gracefully reduce its w. We solve this problem by defining the sending rate updating:

$$R = R + (\alpha - (RTT - RTT_{min}) * R) \times \frac{\beta}{RTT} \tag{4.3}$$

where:

α: target value for backlogged data,

$RTT - RTT_{min}$: the throughput we really get at that time,

$\frac{\beta}{RTT}$: state variable and is maintained as an estimation of the transmission delay of a packet over the network path.

As in TCP Vegas, if $\alpha - (RTT - RTT_{min}) * R < \gamma$, the network path is determined as underutilized and the sending rate will be increased more value. Otherwise, the network path is considered as busy and the sender should gracefully reduce its sending rate.

4.3 Packet Loss

When a packet loss is detected through duplicate ACK or retransmission timeout, the sending rate is

$$R = R \times (1 - 1/\delta) \tag{4.4}$$

In the bandwidth probing state, sending rate is reduced to 0 because the rate is not determined yet in this state. Compare with TCP, we do the same thing as in Multiplicity Decreasing algorithm but adjust the decreasing parameters basing on estimates of queuing delay and buffer size. On loss, the sending rate is decreased as in (4.4). If the RTT is close to the maximal observed value, then the loss is as a buffer overflow of $R \times (1 - 1/2)$, while decreases to $R \times (1 - 1/8)$ as the RTT becomes smaller (as loss is then taken as packet corruption).

Chapter 5

Performance Results

5.1 Performance Metrics

Testing of the simulation results should be measured and analyzed using some performance metrics. In this simulation, the following performance metrics are used: throughput, packet loss rate and delay. We define those metrics in the following formulas.

5.1.1 Throughput

Throughput in cloud datacenters is the average rate of successful message delivery over a communication channel.

$$T_p = \frac{number\ of\ received\ packets}{number\ of\ forwarded\ packets} = \frac{P_a}{P_f} \tag{5.1}$$

where:

T_p : throughput of network.

P_a : packets received over certain time interval

P_f : forwarded packets over certain time interval.

5.1.2 End to end delay

End-to-end delay refers to the time taken for a packet to be transmitted across a network from source to destination.

$$D = packet\ receive\ time - packet\ send\ time = T_d - T_s \tag{5.2}$$

where:

* D : time taken for a packet to be transmitted across a network from the source to the destination node.

* T_d : packet receive time at the destination

* T_s : packet send time at source node

5.1.3 Packet loss rate

The packet loss ratio in cloud datacenters is measured using only end-point packets counter or byte counter data received from ingress and egress network elements:

$$T_d = \frac{number\ of\ packets\ drop}{number\ of\ packets\ drop + number\ of\ packets\ receive} \times 100\% = \frac{P_d}{P_d + P_a} \times 100\% \tag{5.3}$$

where:

T_d: number of packets get dropped before get the destination

P_d: amount of packets drop

P_a : amount of packets received

5.2 Design of simulation structure

In this part we design the general process of simulations. Figure 5.1 shows the overall simulation progress of our work.

* Input packets (traffic sources) and source hosts: We use both long-lived and short-live flows to represent the real traffic in cloud datacenters. A typical distribution that describes the file size is the Pareto with the shaping parameter β between 1 and 2. For $\beta = 1.16$, the Pareto distribution will produce the traffic flows with mean 10KB, minimum size of 1.37KB and median size of 2.5KB. The distribution of the inter-arrival time of new connections is frequently taken to be exponential. In both fat tree and experimental topologies, multiple sources share a common bottleneck edge switch. We define the number of sources as the number of hosts under edge switch. In NS-2 simulation, TCP sources are parameterized by two parameters: the source node and the session number from that node. For each TCP agent, we define a new FTP application. New TCP connections arrive according to a Poisson

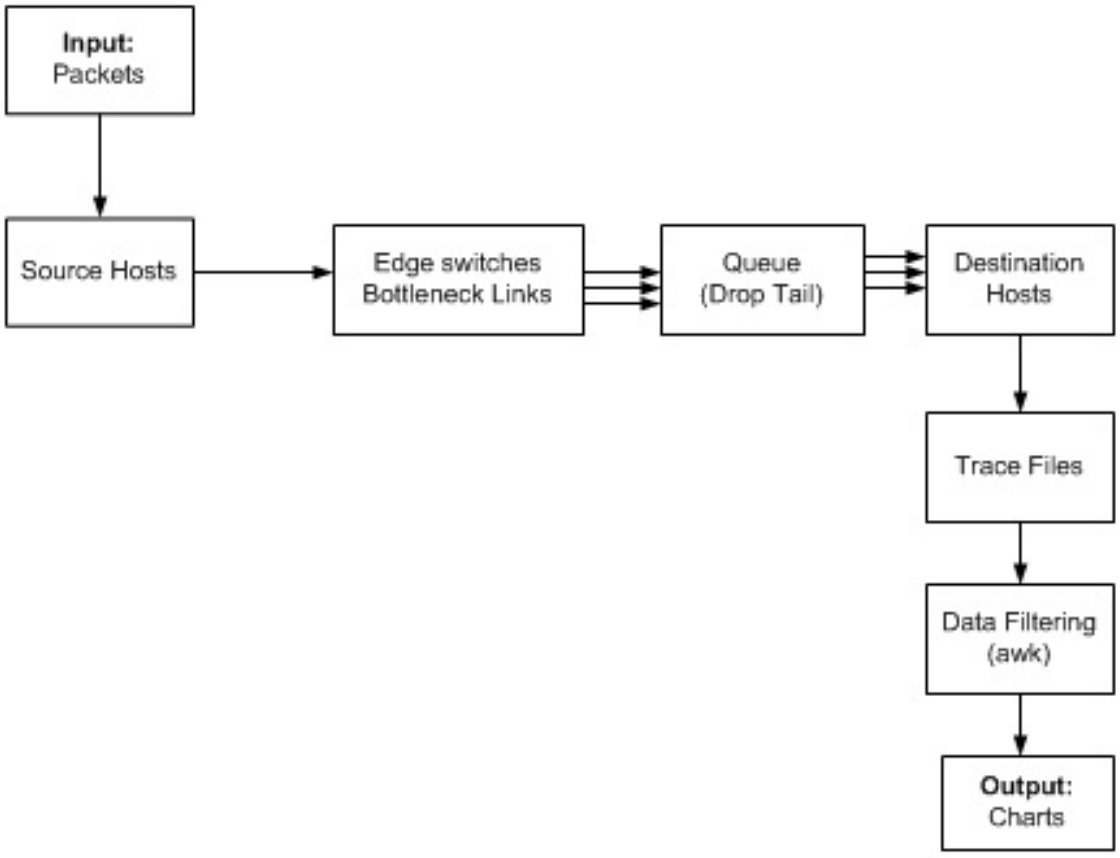


Figure 5.1: Block diagram of the over-all progress of the simulation.

process. We therefore generate the beginning of a new TCP connection using exponentially distributed random variables.

* Edge switches and bottleneck links: To observe the characteristics of the two TCP types. The details are listed in Section 3.1.

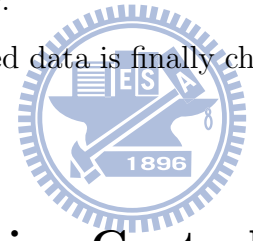
* Queue: We try to create source traffic to flood the network so that all the packets from the source nodes cannot transmit immediately to destination. Thus a Drop-Tail queue system is implemented in the intermediate switches.

* Destination Hosts: Since our main aim is to receive all the packets from the source node to destination node, we create the end of the path with a TCP and proposed sink agent where the results to be collected.

* Trace Files: Here we get all the data from all sources together in a tabular form.

* Data Filtering: Since all packets are collected from all nodes in the network sources filtering should be done here. To do so, AWK programs are used for collecting the sorted data for each TCP type separately.

* Output: Output of the filtered data is finally changed into graph formats and used for analysis.



5.3 Testing Congestion Control Mechanism in the Single Link

We firstly test the proposed mechanism in simple link with and without packet loss. The network model is established in Figure 5.2. Both TCP and ABCC are checked in the same scenario. We wish to run FTP application between each pairs of which the default packet size is 1 KB. Two pairs of hosts are connected by two switches with a bi-directional link that has 5 msec of propagation delay and a capacity of 100 Mbps for each direction. Buffer capacity has the value of 100 packets. In this scenario, we also simulate the error by insert an error model started at the second of 1.5 sec with a random rate in the simple link to evaluate the performance of proposed congestion control in the comparison with TCP.

Figure 5.3 shows the throughput performance of the TCP and ABCC with and without congestion.

As showed in Figure 5.2, without error or no congestion happen, both TCP and ABCC

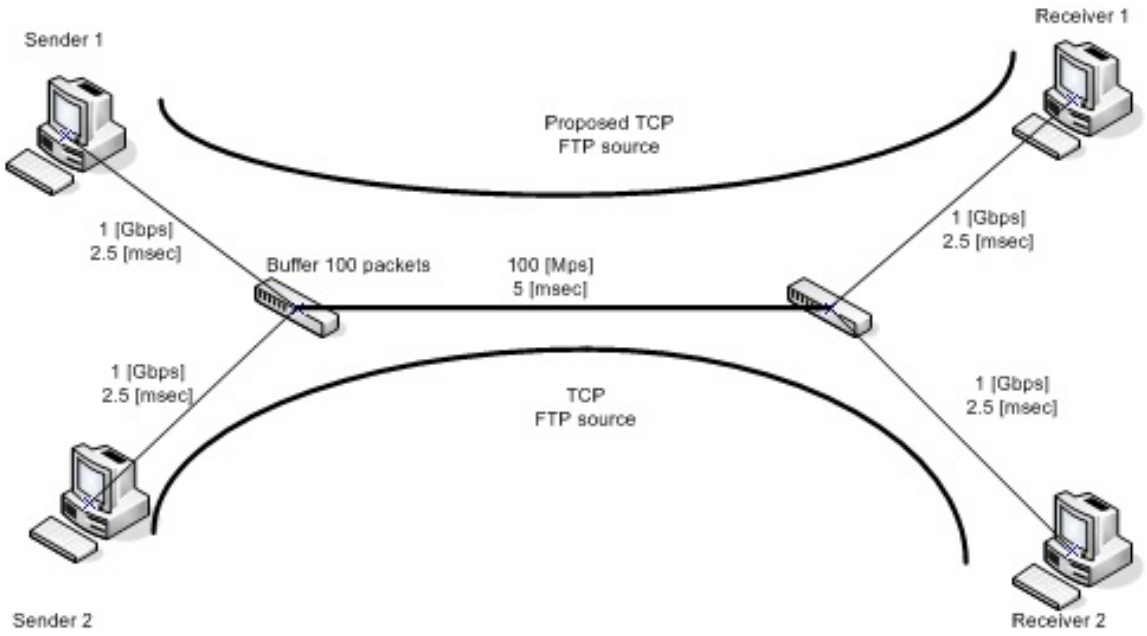


Figure 5.2: Simple link model.

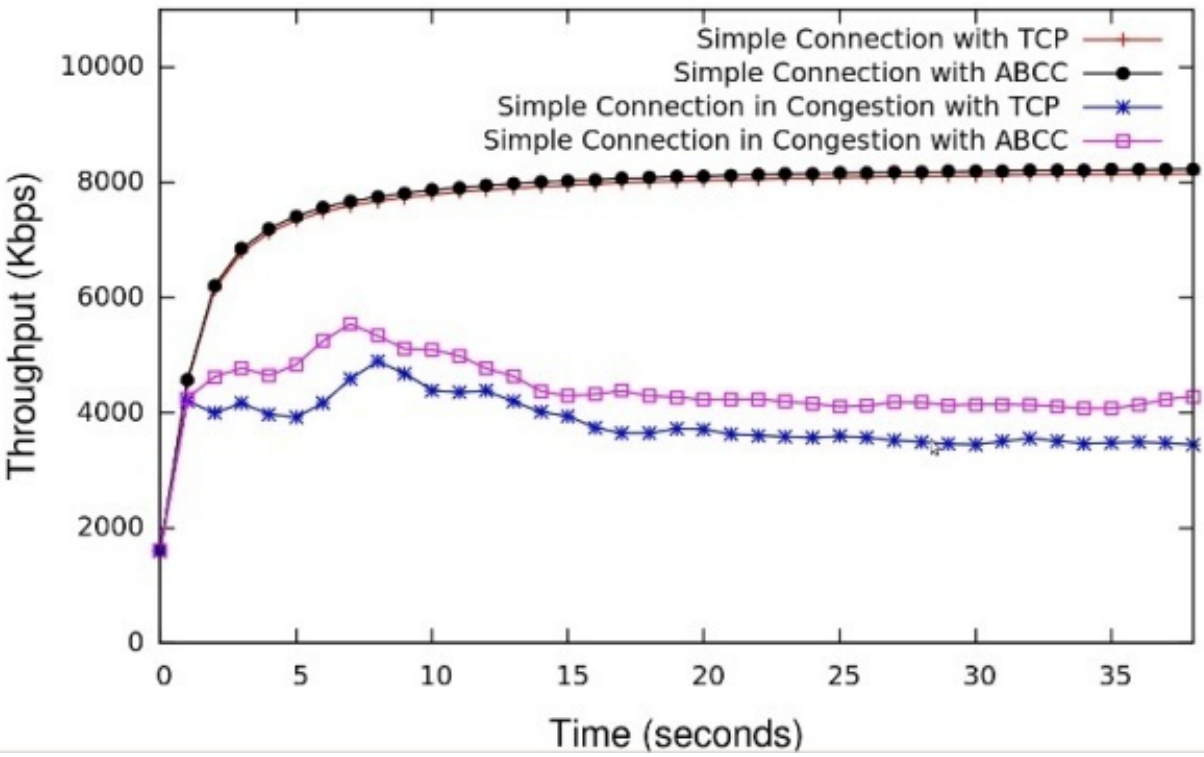


Figure 5.3: Simple link throughput comparison.

mechanisms achieve almost the same throughput during short evaluate time. When congestion happens, indicated by packets loss due to error link, the throughput decreases in both cases. However ABCC has higher throughput than TCP. This simple test proves the effective of proposed mechanism. We will check the performance in details in the following sections.

5.4 Throughput comparisons

5.4.1 Small-scale fattree topology

We use the model that described early in the Section 3.1.1.1 to evaluate the proposed congestion control mechanism in datacenters. To observe the currently used congestion control behaviors of the algorithms, traffic source is created in all-to-all with FTP application between each pairs of which its default packet size is 1KB. Each TCP type is running for 36 sec. Packets sent from TCP sources starting at 0 sec and stops at 36 sec. Figure 5.4 shows the throughput comparisons of single path routing, multipath routing with TCP and ABCC.

Three different models are tested under the same condition. In the single path routing distance vector protocol is chosen in all switches, ECMP with TCP is implemented in NS2 by attaching ECMP ability in all switches, while all the hosts run TCP. We set the same for the ECMP with ABCC, but TCP is replaced by ABCC.

At the beginning of communication, light traffic load created from least load, packets exchange and topology discovery, throughput is the same in three cases: Single Path, ECMP with TCP, ECMP with ABCC. When the traffic in network increases, single path routing has the lowest throughput. The reason is that one path is used and the chosen links transmit traffic load at almost full bandwidth utilization. Making full use of available bandwidth of core links, ECMP with TCP and ECMP with ABCC have the best throughput. In these two models, there are more paths in transmission than flow-based and single path scheme. By the way, both schemes achieve same throughput because no congestion happens in the network. When bottleneck happens in the network, causes by high traffic load is created while there is limited size buffer at switches. Single path has the lowest throughput. Throughput in case ECMP with TCP decreases because there's no effective mechanism for dealing with congestion control. ECMP with ABCC has the best throughput because of

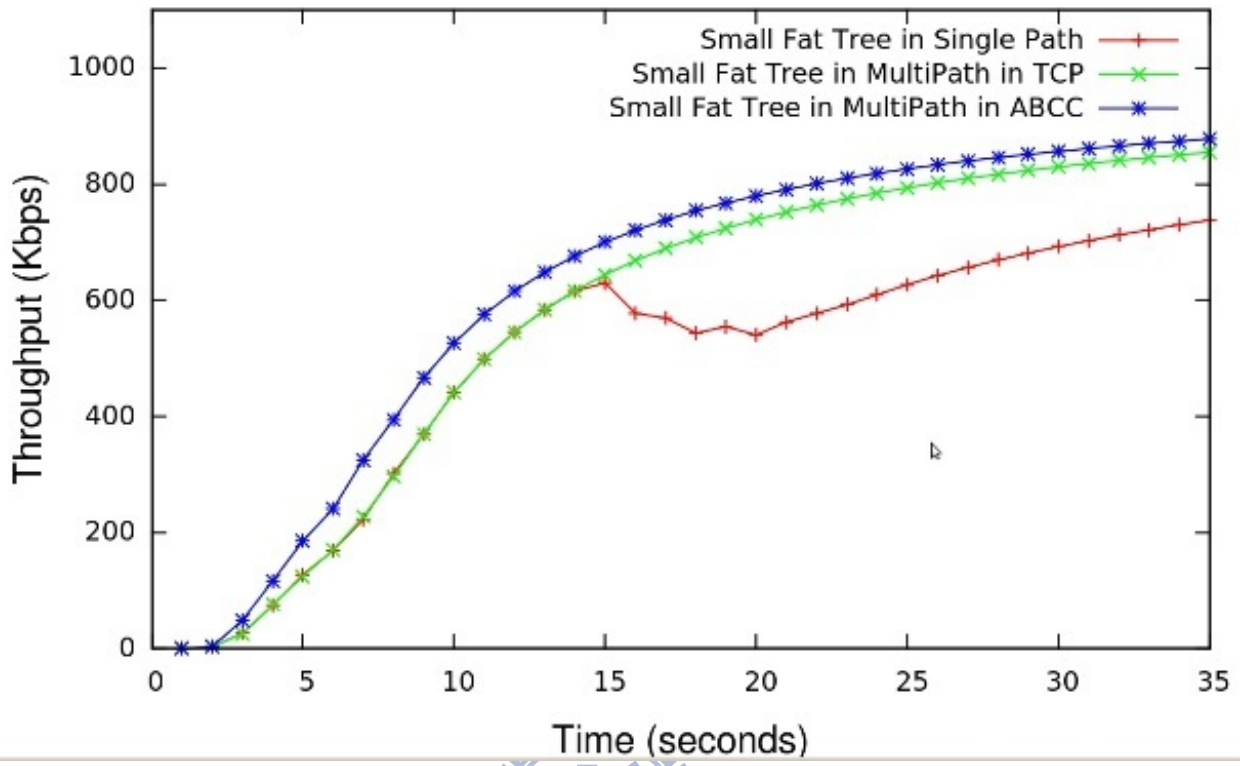


Figure 5.4: Throughput comparison in small-scale fattree topology.

effective congestion control mechanism.

5.4.2 Medium-scale fattree topology

We use the model described early in Section 3.1.1.1 to evaluate the proposed congestion control mechanism in a larger fat tree topology. The network consists of 128 hosts and 80 switches. To observe the currently used congestion control behaviors of the proposed algorithm, traffic source is created as in Section 5.2. Several-to-several traffic is created to observe the congestion phenomenon in the network. All setting in routing and congestion control technique follow the descriptions as in Section 5.4.1.

One of the main and good qualities of ABCC is that it achieves higher throughput than the single path routing and ECMP with TCP as seen from Figure 5.5. The traffic pattern is oscillating due to traffic source creating. As shown in Figure 5.5, ECMP with ABCC gets the peak throughput value far away from other two mechanisms. After estimating the average sending rate, ABCC can achieve the highest bandwidth capacity. TCP just increases its

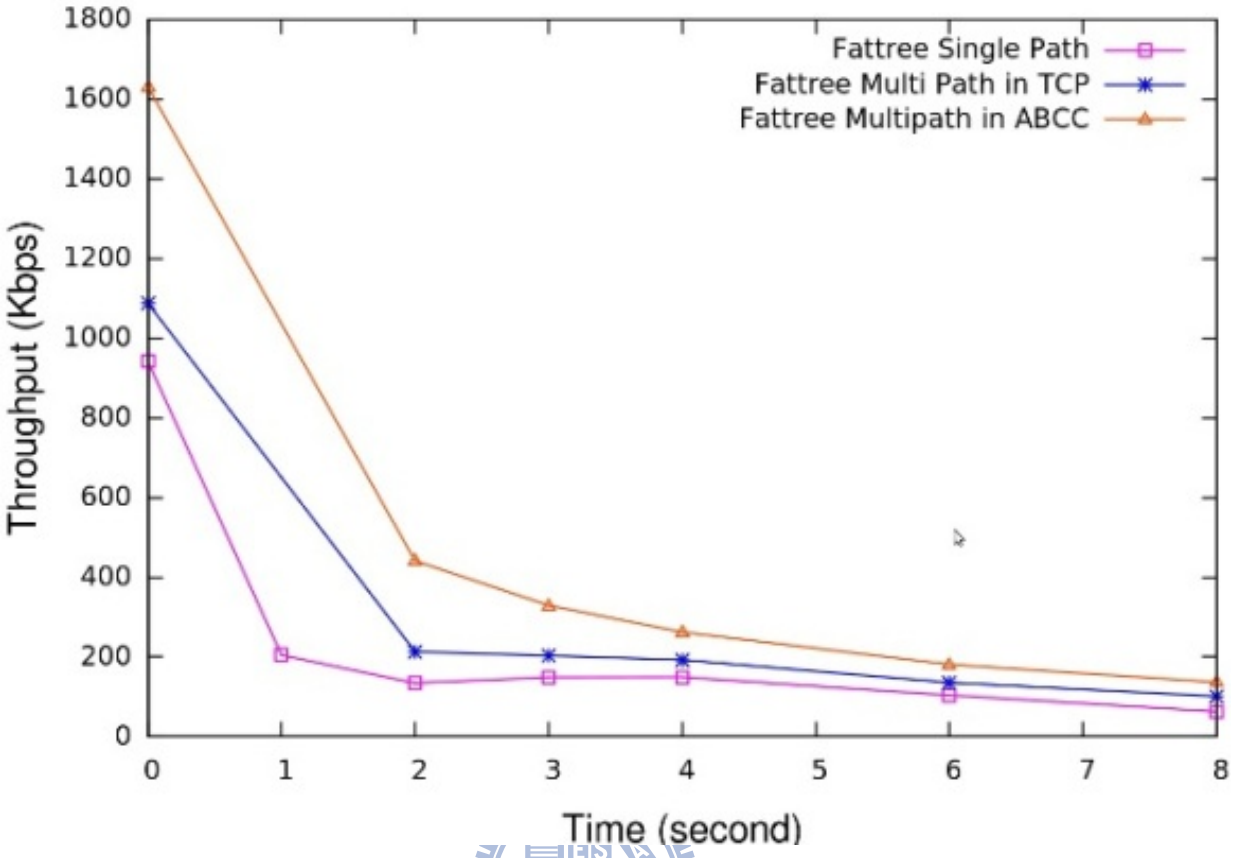


Figure 5.5: Throughput comparison in a medium-scale fattree topology.

window size continuously till the loss of the packets occurs.

5.4.3 Large-scale experimental topology

We use the model descrided early in Section 3.1.1.2 to evaluate the proposed congestion control mechanism in the multipath routing in real cloud datacenter. The system consists of 65 switches and 500 hosts which every 10 hosts under each edge switch. All setting in routing and congestion control technique follow the descriptions as in Section 5.4.1. For diversity the traffic in cloud datacenter, traffic sources from each host include the one which is described in Section 5.2 and several independent long-lived flows; first flow starts at 0 msec, and then, two flows start at 10 msec, 4 flows start at 20 msec, 8 flows start at 30 msec, 16 flows start at 40 msec. At 5 min, all flows except for the first one terminate.

As showed in Figure 5.6 ECMP with ABCC still achieves highest throughput because links transmit traffic at almost full bandwidth utilization. Due to some early sending long-lived flows, the time for achieving the real sending rate is late. Peak value cannot be observed

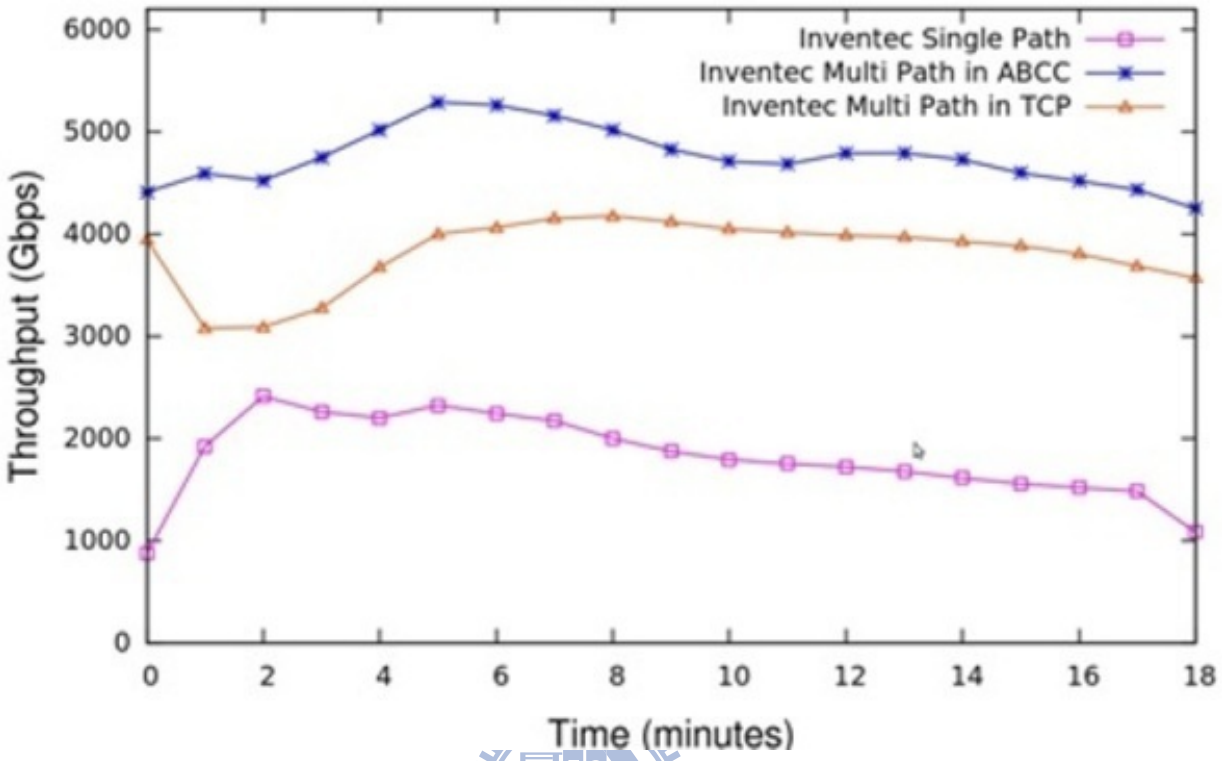


Figure 5.6: Throughput comparison in a large-scale experimental topology.

as in the medium-size fat tree scheme. With additional long-lived flows, the throughput in this scheme is more stable than as in the medium-scale fat tree topology simulation.

5.5 Delay Comparisons

5.5.1 Medium-scale fattree topology

Figure 5.7 shows the delay comparison of the single path and multipath routing in the fat tree topology. Single path has the larger end-to-end delay because the forwarding paths have been in a congestion state and a large number of packets wait in the queue. Multipath routing strategy has the better performance of delay. Figure 5.8 shows the delay comparison of the multipath routing with TCP vs. multipath routing with ABCC. Using adaptive bandwidth scheme and flow-splitting forwarding, multipath with ABCC reduces the delay greatly than single path routing and obtains almost the same performance with multipath with TCP scheme.

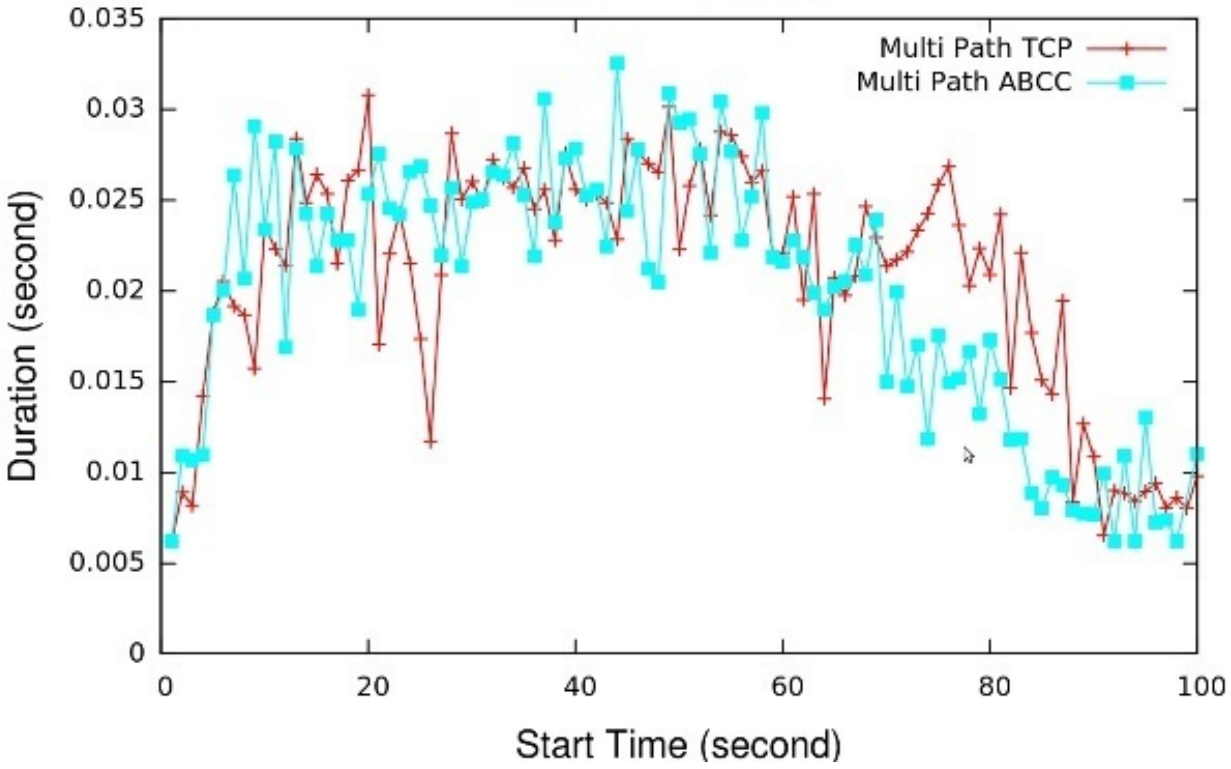


Figure 5.7: End-to-end delay comparisons in fat tree topology using Multipath in TCP vs. Multipath in ABCC

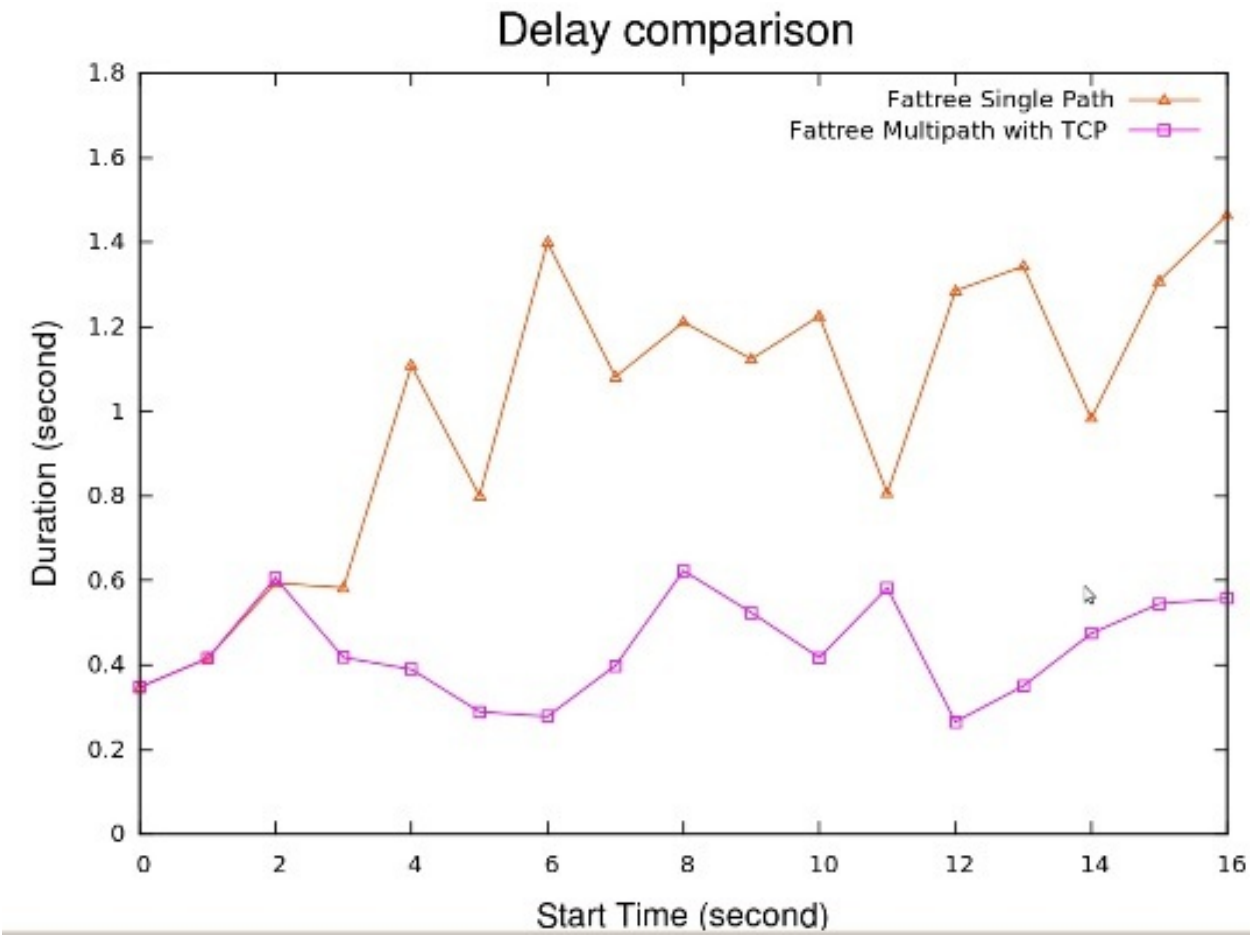


Figure 5.8: End-to-end delay comparisons in fat tree topology using single path vs. multipath routing.

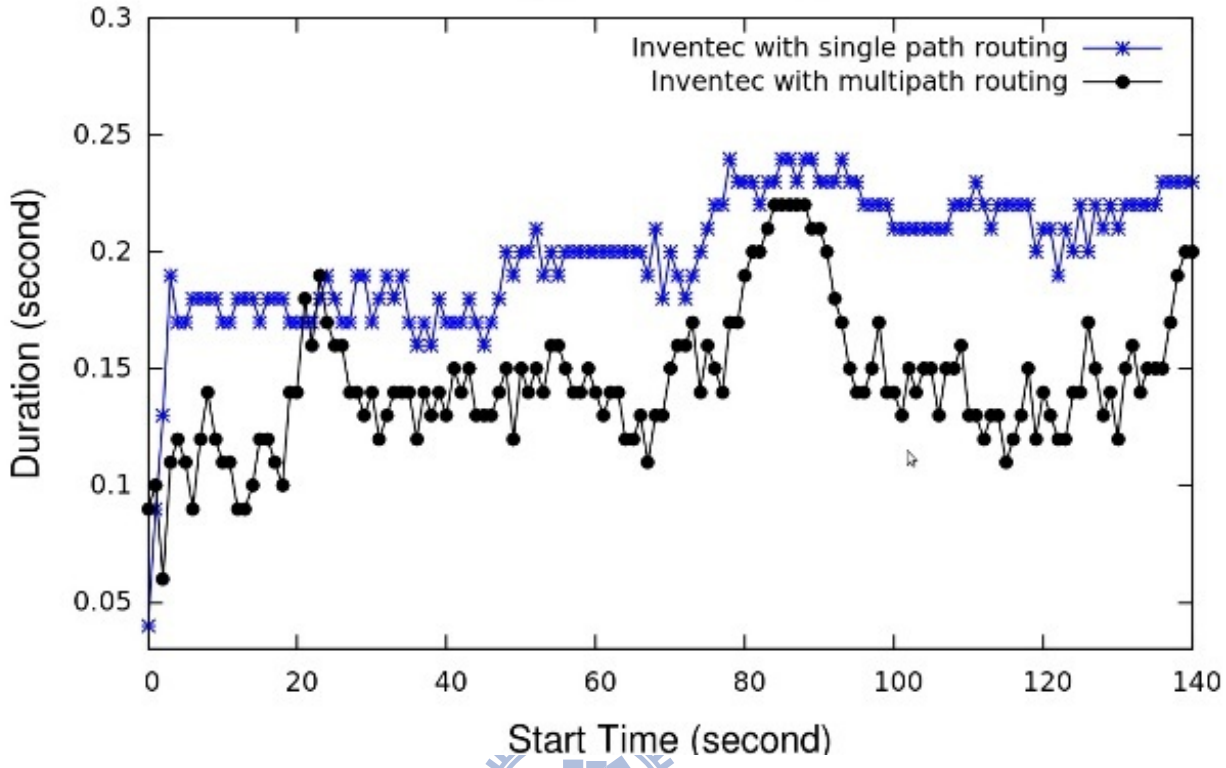


Figure 5.9: End-to-end delay in experimental datacenter using Single Path vs. Multipath.

5.5.2 Large-scale experimental topology

Figures 5.9 and 5.9 show the same results as in the fat tree topology. Only one different thing is the fluctuation. ABCC seems more stable than TCP. The duration for sending a packet from source to destination seems no changing in some interval. When congestion decreases, ABCC tries to send more packets and reaches the queue limit quickly. Once it reaches the limit, it slows down its window and at the same time decreases its number of queued packets. This case is different when using TCP, because TCP never reaches the maximum queue limit. Most of the time TCP keeps very less packet in a queue because TCP's main target is to leave more space free in the queue as much as possible.

5.6 Error rate comparisons

In this step, the number of dropped and received packets are used to observe the behavior of the single path routing and multipath routing with two TCP types when working in the

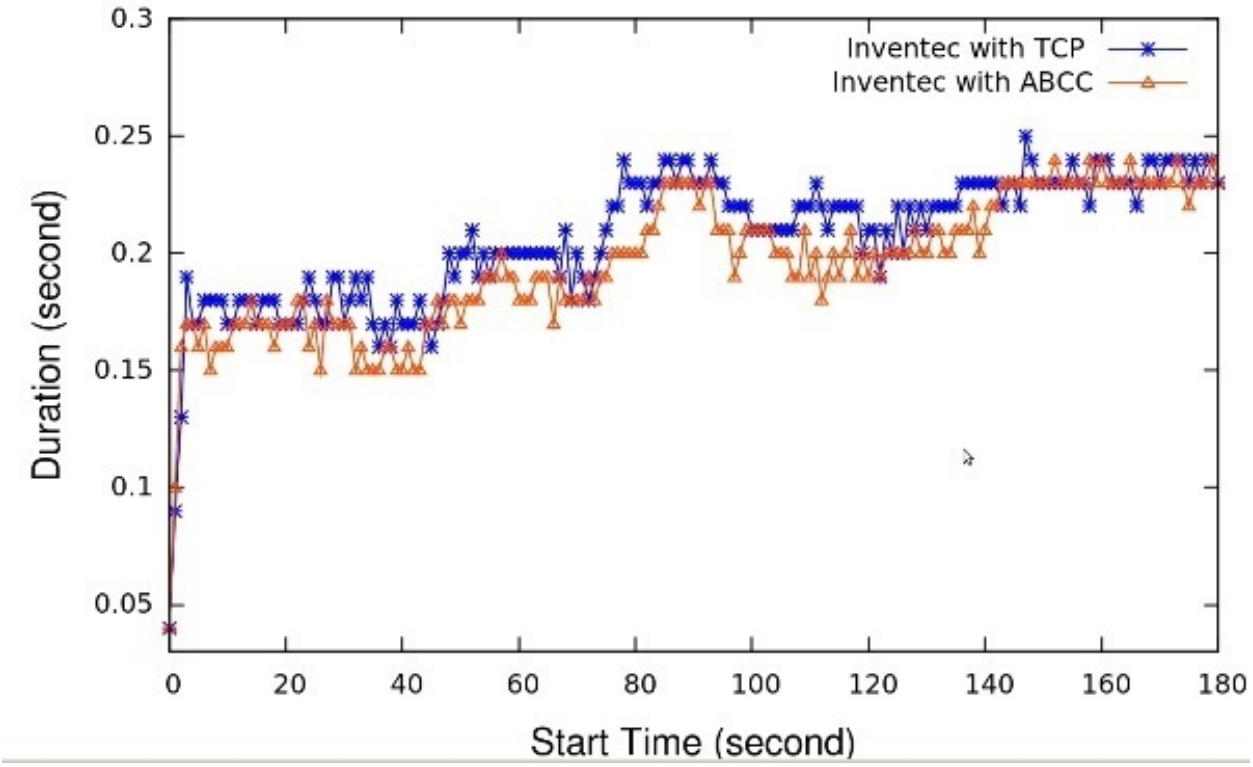


Figure 5.10: End-to-end delay in an experimental datacenter using multipath in TCP vs. multipath in ABCC.

same topology, and the results are displayed in the following tables.

5.6.1 Small-scale fattree topology

The number of the received and dropped packets from the simulation is shown in the Table 5.1. As seen from the table, with flooding traffic, all algorithms are suffering from dropped packets. Single path routing is the weakest mechanism. Because the behaviour of ABCC, it tries to keep updating the sending rate so that ABCC has more ability to face congestion than TCP. TCP has no idea about the available bandwidth so that it continously increases the sending packets and tries to keep many packets in the queue till the buffer is full and a packet is lost. This opposite behaviour challenges TCP and many of its packets are dropped.

Table 5.1: Packet loss rate in small fat tree topology

Type	No.of recv. packets	No.of drop packets	Packet loss rate
Single path	3580	160	4.3%
ECMP with TCP	4012	57	1.4%
ECMP with ABCC	4045	46	1.1%

5.6.2 Large-scale experimental topology

As showed in Table 5.2, ABCC works especially well in the large-scale network. The reason is that the high aggregate bandwidth and the bandwidth adaptive mechanism help ABCC quickly and stably utilize the link capacity even under congestion. ECMP with ABCC reduces the packet loss greater than the single path routing and obtains almost the same performance with TCP.

Table 5.2: Packet loss rate in large-scale experimental topology

Type	No.of recv. packets	No.of drop packets	Packet loss rate
Single path	104565	6783	6.09 %
ECMP with TCP	234567	678	0.29%
ECMP with ABCC	267895	579	0.22%

Chapter 6

Conclusions

In this thesis, we discussed the performance of the cloud datacenter in a multipath scenario under different congestion control mechanisms. Existing protocols for wired networks like TCP and its variants are not suitable for cloud datacenter. Therefore, we concentrated on designing a new TCP variant for high speed and low latency cloud datacenter network.

From the observe results, we find that multipath routing is the good choice for cloud datacenter with any congestion control mechanism. The performance evaluation shows that Adaptive Bandwidth Congestion Control achieves higher efficiency than traditional TCP in the multipath environment. Our proposed ABCC scheme also yields fewer packet retransmissions. This is because of their different congestion avoidance mechanisms. Traditional TCP increases continuously its window size till the loss of the packets occurs. However ABCC has the mechanism for achieving bandwidth available capacity. With real traffic pattern, the end-to-end delay in ECMP with ABCC is more stable than in the condition of fluctuation traffic. ECMP with ABCC works well in the large scale network in the aspect of reducing packet loss.

In the future, as shown in Fig 6.1, we will design a new packet size aware classifier and combine with Adaptive Bandwidth Congestion Control to control the traffic in a size-aware art to get further investigation results.

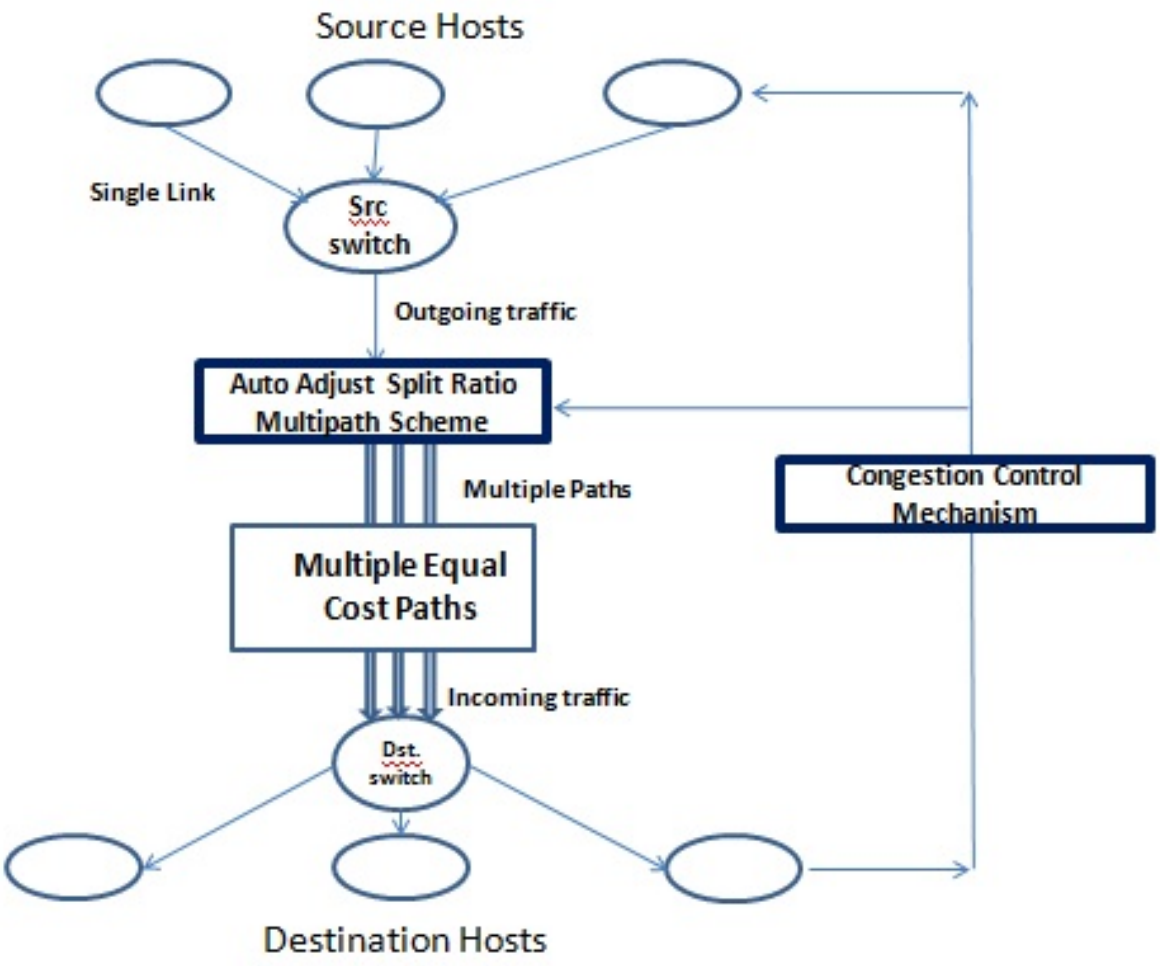


Figure 6.1: Packet-size awared ABCC scheme.

BIBLIOGRAPHY

[**1**] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "BCube: A high performance, server-centric network architecture for modular data centers," ACM SIGCOMM Computer Communication Review, vol. 39, no. 4, pp. 63-74, 2009.

[**2**] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. adhakrishnan, V. Subramanya, and A. Vahdat, "Portland: a scalable fault-tolerant layer 2 data center network fabric," in Proceedings of the ACM SIGCOMM 2009 conference on Data communication, ser. SIGCOMM '09. New York, NY, USA: ACM, 2009, pp. 39-50. [Online]. Available: http://doi.acm.org/10.1145/1592568.1592575.

[**3**] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: a scalable and fault-tolerant network structure for data centers," ACM SIGCOMM Computer Communication Review, vol. 38, no. 4, pp. 75-86, 2008.

[**4**] A. Greenberg, J. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, and S. Sengupta, "VL2: A scalable and flexible data center network," ACM SIGCOMM Computer Communication Review, vol. 39, no. 4, pp. 51-62, 2009.

[**5**] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "Nature of Data-center Traffic: Measurements and Analysis," IMC Internet Measurement Conference, 2009.

[**6**] Sally Floyd, Mark Allman, Amit Jain, and Pasi Sarolahti, "Quick-Start for TCP and IP," RFC 4782. [Online]. Available: http://tools.ietf.org/html/rfc4782.

[**7**] D. Leith, et. al., "Delay-based AIMD congestion control," In PFLDnet 2007, 2007.

[**8**] R. Stewart, Q. Xie, K. Morneault, H. Schwarzbauer, 'Stream Control Transmission Protocol," RFC 2960. [Online]. Available: http://www.ietf.org/rfc/rfc2960.txt.

[**9**] S. Floyd, "HighSpeed TCP for Large Congestion Windows," Network Working Group. [Online]. Available: http://www.ietf.org/rfc/rfc3649.txt.

[**10**] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," SIGCOMM Comput. Commun. Rev., vol. 38, pp. 63-74, August 2008. [Online]. Available: http://doi.acm.org/10.1145/1402946.1402967.

[**11**] E. CHARLES, "Fat-trees: universal networks for hardware-efficient supercomputing," IEEE Transactions on Computers, vol. 34, no. 10, pp. 892-901, 1985.

[**12**] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," SIGCOMM Comput. Commun. Rev., vol. 39, pp. 68-73, 2008. [Online]. Available: http://doi.acm.org/10.1145/1496091.1496103.

[**13**] C. Clos, "A study of non-blocking switching networks," Bell System Technical Journal, 19 vol. 32, no. 2, pp. 406-424, 1953.

[**14**] A. Greenberg, P. Lahiri, D. Maltz, P. Patel, and S. Sengupta, "Towards a next generation datacenter architecture: scalability and commoditization," in Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow. ACM, 2008, pp. 57-62.

[**15**] J. Naous, G. Gibb, S. Bolouki, and N. McKeown, "Netfpga: reusable router architecture for experimental research," in Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow, pp. 1-7, 2008. [Online]. Available: http://doi.acm.org/10.1145/1397718.1397720.

[**16**] D. Thaler and C. Hopps, "Multipath Issues in Unicast and Multicast Next-Hop Selection" RFC 2991 (Informational), Internet Engineering Task Force, Nov. 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2991.txt.

[**17**] M. Kodialam, T. V. Lakshman, and S. Sengupta, "Efficient and Robust Routing of Highly Variable Traffic," IEEE/ACM Transactions on Networking (TON) archive, vol. 17, 2009.

[**18**] L. Han, J. Wang, and C. Wang, "A Novel Multipath Load Balancing Algorithm in Fat-Tree Data Center," CloudCom 2009, LNCS 5931, pp. 405-412, 2009.

[**19**] J. Postel, "Transmission Control Protocol," RFC1122, RFC3168, RFC6093, RFC0793. [Online]. Available: http://www.ietf.org/rfc/rfc793.txt.

[**20**] J. Postel, "Internet Protocol," Advanced Research Projects Agency Information Processing Techniques. [Online]. Available: http://www.ietf.org/rfc/rfc791.txt1981.

[**21**] M. Al-Fares, S. Radhakrishnan, B. Raghavan Nelson Huang, and A. Vahdat, "Hedera: Dynamic Flow Scheduling for Data Center Networks," NSDI'10 Proceedings of the 7th USENIX conference on Networked systems design and implementation USENIX Association Berkeley, CA, USA 2010.

[**22**] Xuan-Yi Lin, Yeh-Ching Chung, and Tai-Yi Huang, "A Multiple LID Routing Scheme for Fat-Tree-Based InfiniBand Networks," 18th International Parallel and Distributed Processing Symposium IPDPS USA 2004.

[**23**] V. Jacobson, "Congestion Avoidance and Control," ACM Computer Communication Review; Proceedings of the Sigcomm '88 Symposium in Stanford, vol. 18, 1988.

[**24**] V. Jacobson, "Re: maximum ethernet throughput...," comp.protocols.tcp-ip Newsgroup. [Online]. Available: http://comp.protocols.tcp-ip Newsgroup.

[**25**] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," [Online]. Available: http://www.ietf.org/rfc/rfc2001.txt.

[**26**] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgement Options," [Online]. Available: http://www.ietf.org/rfc/rfc2018.txt.

[**27**] S. Floyd, S. Ratnasamy, and S. Shenker, "Modifying TCPs Congestion Control for High Speeds," [Online]. Available: http://www.icir.org/floyd/hstcp.html.

[**28**] T. Kelly, "STCP: improving performance in highspeed wide area networks," ACM SIGCOMM Computer Communication Review Homepage archive, vol. 33, 2003.

VITA

Le Thi Lan Huong was born in June 05, 1981 in Quang Binh province, Viet Nam. She received her B.Sc degree from the Electronic and Telecommunication Engineering Department at Danang University of Technology in Vietnam in 2003. From September 2009 to August 2011, she worked on her master degree in the Mobile Computing and Cloud Computing Laboratory of the Department of Communication Engineering at National Chiao-Tung University, Taiwan. Her research interests are in the fields of network design and traffic management in cloud datacenter. You can contact her via e-mail: ltlhuong@gmail.com.