10

Networking for Cloud Computing

Learning Objectives

After studying this chapter, you should be able to

- Understand the general classification of data centers
- Present an overview of the data center environment
- Understand the basic networking issues in data centers
- Explain the performance challenges faced by TCP/IP in data center networks
- Describe the newly designed TCPs for data center networks and their novelty

Preamble

This chapter provides an introduction to networking in Cloud Enabled Data Centers (CEDCs) and the issues thereof. A general classification of data centers and a brief overview of the data center environment are provided to familiarize the reader with the CEDCs. Major issues related to networking in a cloud environment are presented with an emphasis on TCP/IP-related performance issues. Newly designed protocols tailored specifically for data center networks are explained in detail, while mentioning advantages and disadvantages of each.

10.1 Introduction

The Internet over the past few years has transformed from an experimental system into a gigantic and decentralized source of information. Data centers form the backbone of the Internet and host diverse applications ranging

from social networking to web search and web hosting to advertisements. Data centers are mainly classified into two types [1]: the ones that aim to provide online services to users, for example, Google, Facebook, and Yahoo, and others that aim to provide resources to users, for example, Amazon Elastic Compute Cloud (EC2) and Microsoft Azure.

Data centers in the recent past have transformed computing, with large-scale consolidation of enterprise IT into data center hubs and with the emergence of several cloud computing service providers. With the widespread acceptance of cloud computing, data centers have become a necessity. Since cloud computing is becoming an important part of the foreseeable future, studying and optimizing the performance of data centers have become extremely important.

Building an efficient data center is necessary in order to strengthen the data processing and to centrally manage the IT infrastructure. However, ever since the inception of data centers, their operation and maintenance have always been a complex task. It is only after 1994 that the usage of data centers increased extensively. Based on the fault-tolerance capacity and service uptime, today's data centers are classified into four tiers as shown in Table 10.1.

Tier I–IV is a standard methodology used to define the uptime of a data center. This is useful for measuring data center performance, investment, and return on investment (ROI). Tier IV data center is considered to be most robust and less prone to failures. It is designed to host mission critical servers and computer systems with fully redundant subsystems (cooling, power, network links, storage, etc.) and compartmentalized security zones controlled by biometric access control methods. The simplest is Tier I data center, which is usually used by small shops.

As maintaining data centers involves a lot of complexity and cost, small- and medium-range companies cannot build their own data centers. Thus, companies like Google, Amazon, Microsoft, Facebook, and Yahoo transformed into cloud computing service providers and started building Internet data centers (IDCs) to meet the scaling demands of the cloud users.

Today, the data centers of the aforementioned companies host diverse applications such as web search, web hosting, social networking, storage, e-commerce, and large-scale computations. As the variety, complexity, and

TABLE 10.1

Classification of Data Centers

Tiers	Features	Uptime (%)
I	Nonredundant capacity components (single uplinks and servers)	99.671
II	Tier I + redundant capacity components	99.741
III	Tier I + Tier II + dual-powered equipments and multiple links	99.982
IV	Tier I + Tier II + Tier III + all components are fault tolerant including uplinks, storage, HVAC systems, servers + everything is dual powered	99.995

TABLE 10.2

Guide to Where Costs Go in a Data Center

Amortized (%)	Cost Component	Subcomponents
45	Servers	CPU, memory, storage systems
25	Infrastructure	Power distribution and cooling
15	Power draw	Electricity utility costs
15	Network	Links, transit, equipment

Source: Greenberg, A. et al., *SIGCOMM Comput. Commun. Rev.*, 39(1), 68, December 2008.

penetration of such services grow, data centers continue to expand and proliferate. Majority of the data centers, however, face the daunting challenges of reducing the server cost, infrastructure cost, and excessive power consumption and optimizing the network performance. Table 10.2 [2] shows where the costs go in today's cloud service data centers.

This chapter highlights the challenges faced toward designing fast and efficient networks for communication within cloud-enabled data centers (CEDCs). The major emphasis, however, is toward understanding the transport layer issues in data center networks (DCNs).

10.2 Overview of Data Center Environment

Initially, the organizations used to maintain *server rooms*. Generally, server rooms used to house servers and the necessary network electronics for establishing a LAN. Some organizations used to provision one main and one standby server room. The standby server room was equipped to maintain the necessary functions in the event of the main room being put out of action. For better fault tolerance, a few organizations opted to locate these rooms in different buildings.

The advent of client–server computing and the Internet led to the concept of data centers. Large data center companies thrived in the dot-com era when Internet companies faced rapid growth. The data centers were focused on providing reliability. Since then, data center paradigm has served as the foundation for information technology that either runs business or is the business. That paradigm has been evolutional throughout the last several decades and transformational in the past 5–10 years.

The description of a data center has almost always been preceded with *mission critical*, because that is the service it provides—the mission critical hardware and software where maximum uptime is required. The data center is a fortress, dedicated to achieving maximum reliability at any cost. While reliability is still the key factor, the data center has evolved and advancements in the past 5 years have accelerated the pace of innovation.

10.2.1 Architecture of Classical Data Centers

The data center is home to the computational power, storage, and applications necessary to support an enterprise business. The data center infrastructure is central to the IT architecture, from which all content is sourced or passes through. Proper planning of the data center infrastructure design is critical, and performance, resiliency, and scalability need to be carefully considered.

The *multitier* model is the most common design in the enterprise. It is based on the web, application, and database layered design supporting commerce and enterprise business ERP and CRM solutions. This type of design supports many web service architectures, such as those based on Microsoft. NET or Java 2 Enterprise Edition. These web service application environments are used by ERP and CRM solutions from Siebel and Oracle, to name a few. The multitier model relies on security and application optimization services to be provided in the network.

The *server cluster* model has grown out of the university and scientific community to emerge across enterprise business verticals including financial, manufacturing, and entertainment. The server cluster model is most commonly associated with high-performance computing (HPC), parallel computing, and high-throughput computing (HTC) environments but can also be associated with grid/utility computing. These designs are typically based on customized, and sometimes proprietary, application architectures that are built to serve particular business objectives.

Multitier model: The multitier data center model is dominated by HTTP-based applications in a multitier approach. The multitier approach includes web, application, and database tiers of servers. Today, most web-based applications are built as multitier applications. The multitier model uses software that runs as separate processes on the same machine using interprocess communication (IPC), or on different machines with communications over the network. Typically, the following three tiers are used:

1. Web server
2. Application
3. Database

Multitier server farms built with processes running on separate machines can provide improved resiliency and security. Resiliency is improved because a server can be taken out of service while the same function is still provided by another server belonging to the same application tier. Security is improved because an attacker can compromise a web server without gaining access to the application or database servers. Web and application servers can coexist on a common physical server; the database typically remains separate.

10.2.2 CEDCs

New technological and economic pressures have forced organizations to look at ways to get more out of their IT infrastructure. The current state of IT infrastructure is strained, and the new demands make it all the more difficult for businesses to maintain efficiency and effectiveness. This has a bearing on the quality of services irrespective of the number of users and applications. The solution lies in cloud computing. The cloud has the capability to reduce costs and increase the flexibility of applications and services including IT infrastructure. The CEDC takes a virtualized data center (and business) into one that is more agile because it takes virtualization to the next level. The virtualized environment is transformed into one that is optimized with intelligent, integrated IaaS, and PaaS that manages dynamic workloads—giving the workloads the resources that it needs based on business policies. It also provides automation and orchestration of resources across heterogeneous data centers. This progression from a virtualization management to a CEDC is important as it addresses common business goals we hear over and over again.

10.2.3 Physical Organization

Data centers generally span across the entire building, a few floors of a building or even a single room in the building. Figure 10.1, collected from Google images, shows one of the several possible ways of arranging the server racks in a data center. Data centers usually comprise of a large number of servers that are mounted in rack cabinets and are placed in single rows forming corridors (so-called aisles) between them, so as to allow access to the front and rear of each cabinet.

FIGURE 10.1
Physical organization of a data center.

Moreover, a few equipments such as storage devices are often as large as the racks. Such equipments are generally placed alongside the racks. Large data centers house several thousand servers wherein sometimes shipping containers packed with 1000 or more servers each are used. In the event of a failure or when upgrades are required, the entire containers are replaced rather than replacing an individual server.

10.2.4 Storage and Networking Infrastructure

Typically, data centers require four different types of network accesses and, hence, could use four different types of physical networks as shown below:

1. *Client–server network*: To provide external connectivity to the data center. Traditional wired Ethernet or Wireless LAN technologies can be used.
2. *Server–server network*: To provide high-speed communication among the servers of the data center. Ethernet, InfiniBand (IBA), or other technologies can be used. Figure 10.2 shows an example of server–server network.
3. *Server–storage network*: To provide high-speed connectivity between the servers and storage devices. Usually Fiber Channel is used, but technologies like Ethernet or InfiniBand can also be used.
4. *Other networks* such as a network required to manage the data center. Generally, Ethernet is used but the cabling may be different from the mainstream networks.

Multiple applications run inside a single data center, typically with each application hosted on its own set of (potentially virtual) server machines.

FIGURE 10.2
Networking infrastructure in data centers.

Each application is associated with one or more publicly visible and routable IP addresses to which clients in the Internet send their requests and from which they receive replies. Inside the data center, requests are spread among a pool of frontend servers that process the requests. This spreading is typically performed by a specialized load balancer.

A two-tier network topology is very popular in DCNs today. Access switches for server connectivity are collapsed in high-density aggregation switches that provide the switching and routing functionalities for access switching interconnections and various LAN servers. It has several benefits:

- Design simplicity (fewer switches and so fewer managed nodes)
- Reduced network latency (by reducing the number of switch hops)
- Typically a reduced network design oversubscription ratio
- Lower aggregate power consumption

However, a disadvantage of a two-tier design includes limited scalability: when the ports on an aggregation switch pair are fully utilized, then the addition of another aggregation switch/router pair adds a high degree of complexity. The connection between aggregation switch pairs must be fully meshed with high bandwidth, so no bottlenecks are introduced into the network design. Since an aggregation switch pair is also running routing protocols, more switch pairs means more routing protocol peering and more routing interfaces and complexity introduced by a full mesh design.

10.2.5 Cooling Infrastructure

Since the data centers usually span across the entire building and house several thousand servers, a sophisticated cooling infrastructure is deployed, which may involve building level air-conditioning units, fans, and air recirculation systems. Figure 10.3 shows one of the possible arrangements of aisles by which the cooling infrastructure is simplified.

The server racks are placed on a raised plenum and arranged in alternately back-facing and front-facing aisles. Cold air is forced up in the front-facing aisles, and the server or chassis fans draw the cold air through the server to the back. The hot air on the back then rises and is directed (sometimes by using some deflectors) toward the chiller plant for cooling and recirculation. The author in [3] mentions that although such a setup is not expensive, it can create hot spots either due to uneven cooling or the mixing of hot and cold air.

In recent years, there have been many innovations in power and cooling technologies and management of facilities. Efficiencies have been integrated into every aspect of the data center and building design, covering everything

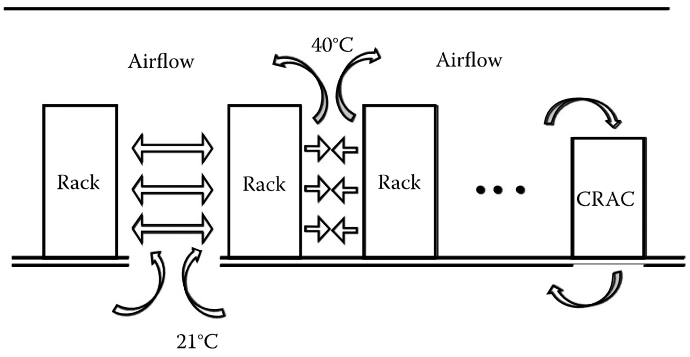

FIGURE 10.3
Cooling in a data center. (From Kant, K., *Comput. Netw.*, 53(17), 2939, 2009.)

from bunkers to chicken coop design and mobile data centers to using the building as an air handler. Green technologies and environmental awareness have also been a large part of the industry in the past 3 years. No longer just a choice between build and lease, the data center can be owned, placed in colocation, wholesale, put in a public or private cloud, or a hybrid strategy of options.

The changes that have come about have even altered the meaning of what constitutes a data center. To Google, Microsoft, or Yahoo, it is a hyperscale facility with tremendous innovation engineered into it. For the consolidation projects, it means taking what they once considered to be data centers and bringing them into a small number of new, large-scale facilities. To others, their definition of a data center was transformed by the advances in IT equipment that required more power, more cooling, and a more advanced facility to support it.

Many facets of site selection for data centers were shaped by the proliferation of fiber networks and the need to both avoid natural disasters and achieve extreme power and cooling efficiencies. In the United States, several regions became data center hubs, where Internet companies and enterprises are selected to build new data centers. Silicon Valley continued to prosper and grow, and regional hubs developed in Quincy, Washington, Chicago, Dallas/Fort Worth, North Carolina, and the New York/New Jersey region.

10.2.6 Nature of Traffic in Data Centers

Data center environment is largely different from that of the Internet, for example, the round-trip time (RTT) in DCNs can be as less as 250 μs in the absence of queuing [4]. The reason is that DCNs are privately owned networks tailored to achieve high bandwidth and low latency. Moreover,

TABLE 10.3

Data Center Traffic: Applications and Performance Requirements

Traffic Type	Examples	Requirements
Mice traffic (<100 kB)	Google search, Facebook	Short response times
Cat traffic (100 kB to 5 MB)	Picasa, YouTube, Facebook photos	Low latency
Elephant traffic (>5 MB)	Software updates, video on demand	High throughput

Source: Reproduced from Kant, K., *Comput. Netw.*, 53(17), 2939, December 2009.

the nature of traffic in DCNs largely varies from that of the Internet traffic. Traffic in DCNs is classified mainly into three types [5]:

1. *Mice traffic*: The queries form the mice traffic (e.g., Google search and Facebook updates). Majority of the traffic in a DCN is query traffic, and its data transmission volume is usually less.
2. *Cat traffic*: The control state and coordination messages form the cat traffic (e.g., small- and medium-sized file downloads)
3. *Elephant traffic*: The large updates form the elephant traffic (e.g., anti-virus updates and movie downloads).

The different traffic types in DCNs, their applications, and performance requirements are summarized in Table 10.3. Thus, bursty query traffic, delay-sensitive cat traffic, and throughput-sensitive elephant traffic coexist in DCNs.

10.3 Networking Issues in Data Centers

Cloud networking is the fundamental backbone to provide cloud services and the reason behind the shift in how IT services are provided to users. This section focuses on several issues that are faced in cloud networking.

10.3.1 Availability

One of the daunting challenges that a cloud provider organization faces is to provide maximum uptime for the services that are offered to the users. Even a few seconds of downtime may lead to loss of reputation for the organization and affect the overall business. Moreover, downtime may lead to violation of service-level agreements (SLAs) between the cloud user and the cloud provider, thus largely affecting the cloud provider's revenues. The most widely adopted approach to achieve high availability is to replicate the data and take regular backups.

10.3.2 Poor Network Performance

The three basic performance requirements of a DCN are high burst tolerance, low latency, and high throughput [5]. This is because traffic in a data center comprises a mix of all the three kinds of traffic: mice traffic, cat traffic, and elephant traffic, each having a different performance requirement than the other.

The major challenge is that traditional Transmission Control Protocol/ Internet Protocol (TCP/IP) stack, which is mainly designed for Internet-like scenarios, fails to provide optimal performance in DCNs. The next section provides a more detailed description about several TCP/IP issues in DCNs.

10.3.3 Security

Keeping a cloud user's data secure during transit, or while it is at rest, is yet a concern for the cloud service providers. Accidental deletion of data because of a power outage or malfunctioning of a regular backup program may lead to loss of customer's data and incur huge damage for the hosting organization.

Apart from the aforementioned concerns, cloud providers also need to ensure physical security of a data center building and its networking infrastructure to prevent any attacks from malicious insiders.

10.4 Transport Layer Issues in DCNs

TCP is one of the most dominant transport protocols, widely used by a large variety of Internet applications, and also constitutes majority of the traffic in both types of DCNs [5]. It has been the workhorse of the Internet ever since its inception. The success of the Internet, in fact, can be partly attributed to the congestion control mechanisms implemented in TCP. Though the scale of the Internet and its usage increased exponentially in the recent past, TCP has evolved to keep up with the changing network conditions and has proven to be scalable and robust.

However, it has been observed that the state-of-the-art TCP fails to satisfy the three basic requirements (mentioned in the previous subsection) together within the time boundaries because of impairments such as TCP incast [4], TCP outcast [6], queue buildup [5], buffer pressure [5], and pseudocongestion effect [7], which are discussed further in the following sections.

10.4.1 TCP Impairments in DCNs

Although TCP constantly evolved over a period of three decades, the diversity in the characteristics of present and next-generation networks and a variety of application requirements have posed several challenges to TCP

congestion control mechanisms. As a result, the shortcomings in the fundamental design of TCP have become increasingly apparent. In this section, we mainly focus on the challenges faced by the state-of-the-art TCP in DCNs.

10.4.1.1 TCP Incast

TCP incast has been defined as the pathological behavior of TCP that results in gross underutilization of the link capacity in various many-to-one communication patterns [8], for example, partition/aggregate application pattern as shown in Figure 10.4. Such patterns are the foundation of numerous large-scale applications like web search, MapReduce, social network content composition, and advertisement selection [5,9]. As a result, TCP incast problem widely exists in today's data center scenarios such as distributed storage systems, data-intensive scalable computing systems, and partition/aggregate workflows [1].

In many-to-one communication patterns, an aggregator issues data requests to multiple worker nodes. The worker nodes, upon receiving the request, concurrently transmit a large amount of data to the aggregator (see Figure 10.5). The data from all the worker nodes traverse a bottleneck link in many-to-one fashion. The probability that all the worker nodes send the reply at the same time is high because of the tight time bounds. Therefore, the packets from these nodes happen to overflow the buffers of top-of-the-rack (ToR) switches and, thus, lead to packet losses. This phenomenon is known as synchronized

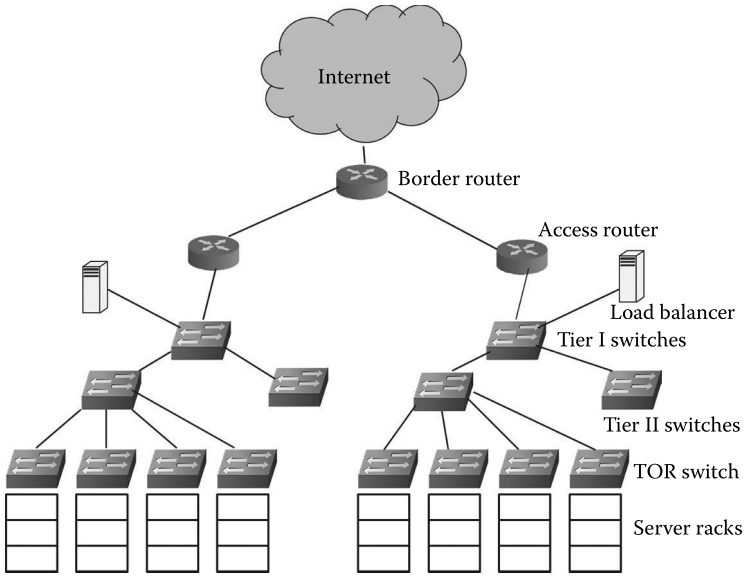

FIGURE 10.4
Partition/aggregate application structure. (From Kurose, J.F. and Ross, K.W., *Computer Networking: A Top Down Approach*, 6th edn., Addison-Wesley, 2012.)

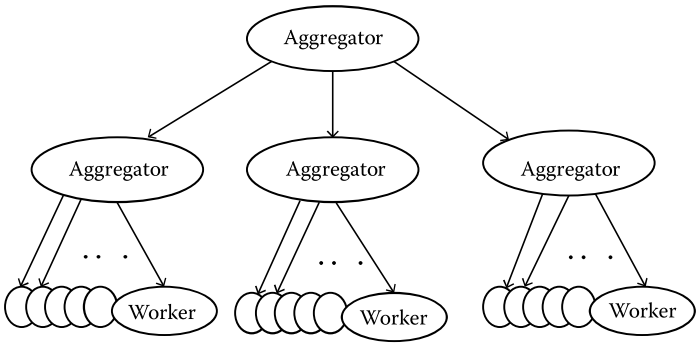

FIGURE 10.5
TCP incast.

mice collide [5]. Moreover, no worker node can transmit the next data block until all the worker nodes finish transmitting the current data block. Such a transmission is termed as barrier synchronized transmission [9].

Under such constraints, as the number of concurrent worker nodes increases, the perceived application level throughput at the aggregator decreases due to a large number of packet losses. The lost packets are retransmitted only after the retransmit timeout (RTO), which is generally in the order of a few *milliseconds*. As mentioned earlier, mice traffic requires short response time and is highly delay sensitive. Frequent timeouts resulting from incast significantly degrade the performance of mice traffic as the lost packets are retransmitted after a few *milliseconds*.

It must be noted that a *fast retransmit* mechanism may not be applicable to mice traffic applications since the data transmission volume of such traffic is quite less, and hence, there are very few packets in the entire flow. As a result, the sender (or worker node) may not get sufficient duplicate acknowledgements (*dupacks*) to trigger a fast retransmit.

Mitigating TCP incast: A lot of solutions, ranging from application layer solutions to transport layer solutions and link layer solutions, have been proposed recently to overcome the TCP incast problem. A few solutions suggest *revision of TCP*, others recommend to *replace TCP*, while some seek solutions from layers other than the transport layer to solve this problem [1]. Ren et al. [11] provides a detailed analysis and summary of all such solutions.

10.4.1.2 TCP Outcast

When a large set of flows and a small set of flows arrive at two different input ports of a switch and compete for the same bottleneck output port, the small set of flows lose out on their throughput share significantly. This phenomenon has been termed as TCP outcast [6] and mainly occurs in data center switches that employ drop-tail queues. Drop-tail queues lead to consecutive packet drops from one port and, hence, cause frequent TCP timeouts. This property

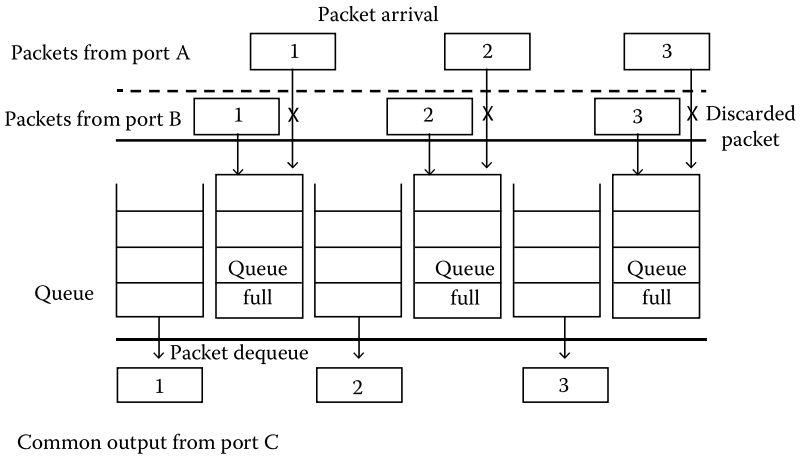

FIGURE 10.6
Example scenario of port blackout. (From Prakash, P. et al., The TCP outcast problem: Exposing unfairness in data center networks, in *Proceedings of the Ninth USENIX Conference on Networked Systems Design and Implementation*, ser. *NSDI'12*, USENIX Association, Berkeley, CA, 2012, pp. 30–30, available [Online]: http://dl.acm.org/citation.cfm?id=2228298.2228339.)

of drop-tail queues is termed as *port blackout* [6], and it significantly affects the performance of small flows because frequent timeouts lead to high latencies and, thus, poor-quality response times. Figure 10.6 shows an example scenario of a port blackout, where A and B are input ports and C is the common output port. The figure shows that packets arriving at port B are successfully queued whereas those arriving at port A are dropped consecutively.

It is well known that the throughput of a TCP flow is inversely proportional to the RTT of that flow. This behavior of TCP leads to RTT bias, that is, flows with low RTT achieve larger share of bandwidth than the flows with high RTT. However, it has been observed that due to TCP outcast problem in DCNs, TCP exhibits *inverse RTT bias* [6], that is, flows with low RTT are outcasted by flows with high RTT.

The two main factors that cause TCP outcast are (1) the usage of drop-tail queues in switches and (2) many-to-one communication pattern, which leads to a large set of flows and a small set of flows arriving at two different input ports and competing for the same bottleneck output port. Both these factors are quite common in DCNs since majority of the switches employ drop-tail queues and many-to-one communication pattern is the foundation of many cloud applications.

Mitigating TCP outcast: A straightforward approach to mitigate TCP outcast is to use queuing mechanisms other than drop tail, for example, random early detection (RED) [12] and stochastic fair queue (SFQ) [6]. Another possible approach is to minimize the buffer occupancy at the switches by designing efficient TCP congestion control laws at the end hosts.

10.4.1.3 Queue Buildup

Due to the diverse nature of cloud applications, mice traffic, cat traffic, and elephant traffic coexist in a DCN. The long-lasting and greedy nature of elephant traffic drives the network to the point of extreme congestion and overflows the bottleneck buffers. Thus, when both mice traffic and elephant traffic traverse through the same route, the performance of mice traffic is significantly affected due to the presence of the elephant traffic [5].

Following are two ways in which the performance of mice traffic is degraded due to the presence of elephant traffic [5]: (1) Since most of the buffer is occupied by elephant traffic, there is a high probability that the packets of mice traffic get dropped. The impact of this situation is similar to that of TCP incast because the performance of mice traffic is largely affected by frequent packet losses and, hence, the timeouts. (2) The packets of mice traffic, even when none are lost, suffer from increased queuing delay as they are in queue behind the packets of elephant traffic. This problem is termed as queue buildup.

Mitigating queue buildup: Queue buildup problem can be solved only by minimizing the queue occupancy in the DCN switches. Most of the existing TCP variants employ reactive approach toward congestion control, that is, they do not reduce the sending rate unless a packet loss is encountered and, hence, fail to minimize the queue occupancy. A proactive approach instead is desired to minimize the queue occupancy and overcome the problem of queue buildup.

10.4.1.4 Buffer Pressure

Buffer pressure is yet another impairment caused by the long-lasting and greedy nature of elephant traffic. When both mice traffic and elephant traffic coexist on the same route, most of the buffer space is occupied by packets from the elephant traffic. This leaves a very little room to accommodate the burst of mice traffic packets arising out of many-to-one communication pattern. The result is that a large number of packets from mice traffic are lost, leading to poor performance. Moreover, majority of the traffic in DCNs is bursty [5], and hence, packets of mice traffic get dropped frequently because the elephant traffic lasts for a longer time and keeps most of the buffer space occupied.

Mitigating buffer pressure: Like queue buildup, buffer pressure problem too can be solved by minimizing the buffer occupancy in the switches.

10.4.1.5 Pseudocongestion Effect

Virtualization is one of the key technologies driving the success of cloud computing applications. Modern data centers adopt virtual machines (VMs) to offer on-demand cloud services and remote access. These data centers are known as *virtualized data centers* [1,7]. Though there are several advantages of virtualization like efficient server utilization, service isolation, and low system maintenance cost [1], it significantly affects the environment where

TABLE 10.4

TCP Impairments in DCNs and Their Causes

TCP Impairment	Causes
TCP incast	Shallow buffers in switches and bursts of mice traffic resulting from many-to-one communication pattern
TCP outcast	Usage of tail-drop mechanism in switches
Queue buildup	Persistently full queues in switches due to elephant traffic
Buffer pressure	Persistently full queues in switches due to elephant traffic and bursty nature of mice traffic
Pseudocongestion effect	Hypervisor scheduling latency

our traditional protocols (e.g., TCP and UDP [user datagram protocol]) work. The recent study of Amazon EC2 data center reveals that virtualization dramatically deteriorates the performance of TCP and UDP in terms of both throughput and end-to-end delay [1]. Throughput becomes unstable, and the end-to-end delay becomes quite large even if the network load is less [1].

When more number of VMs are running on the same physical machine, the hypervisor scheduling latency increases the waiting time for each VM to obtain an access to the processor. Hypervisor scheduling latency varies from microseconds to several hundred milliseconds [7], leading to unpredictable network delays (i.e., RTT) and, thus, affecting the throughput stability and largely increasing the end-to-end delay. Moreover, hypervisor scheduling latency can be so high that it may lead to RTO at the VM sender. Once RTO occurs, VM sender assumes that the network is heavily congested and significantly brings down the sending rate. We term this effect as *pseudocongestion effect* because the congestion sensed by the VM sender is actually *pseudocongestion* [7].

Mitigating pseudocongestion effect: There are generally two possible approaches to address the aforementioned problem. One is to design efficient schedulers for hypervisor so that the scheduling latency can be minimized. Another approach is to modify TCP such that it can intelligently detect the *pseudocongestion* and react accordingly.

10.4.2 Summary: TCP Impairments and Causes

We briefly summarize the TCP impairments discussed in the previous subsections and mention the causes for the same in Table 10.4.

10.5 TCP Enhancements for DCNs

Recently, a few TCP variants, specifically data transport, have been proposed in DCNs. The major goal of these TCP variants is to overcome the aforementioned impairments and improve the performance of TCP in DCNs.

This chapter presents the background and the causes of each of the aforementioned impairments, followed by a comparative study of TCP variants that aim to overcome these impairments. Although a few other transport protocols have also been proposed for DCNs, we restrict the scope of this chapter to TCP variants because TCP is the most widely deployed transport protocol in modern operating systems.

10.5.1 TCP with Fine-Grained RTO (FG-RTO) [4]

The default value of minimum RTO in TCP is generally in the order of milliseconds (around 200 ms). This value of RTO is suitable for Internet-like scenarios where the average RTT is in the order of hundreds of milliseconds. However, it is significantly larger than the average RTT in data centers, which is in the order of a few microseconds. A large number of packet losses due to TCP incast, TCP outcast, queue buildup, buffer pressure, and pseudocongestion effect result in frequent timeouts and, in turn, lead to missed deadlines and significant degradation in the performance of TCP. Vasudevan et al. [4] show that reducing the minimum RTO from 200 ms to 200 μs significantly alleviates the problems of TCP in DCNs and improves the overall throughput by several orders of magnitude.

Advantages: The major advantage of this approach is that it requires minimum modification to the traditional working of TCP and thus can be easily deployed.

Shortcomings: The real-time deployment of fine-grained timers is a challenging issue because the present operating systems lack the high-resolution timers required for such low RTO values. Moreover, FG-RTOs may not be suitable for servers that communicate to clients through the Internet. Apart from the implementation issues of fine-grained timers, it must be noted that this approach of eliminating drawbacks of TCP in DCNs is a *reactive* approach. It tries to reduce the impact of a packet loss rather than *avoiding* the packet loss in the first place. Thus, although this approach significantly improves the network performance by reducing post–packet loss delay, it does not alleviate the TCP incast problem for loss-sensitive applications.

10.5.2 TCP with FG-RTO + Delayed ACKs Disabled [4]

Delayed ACKs are mainly used for reducing the overhead of ACKs on the reverse path. When delayed ACKs are enabled, the receiver sends only one ACK for every two data packets received. If only one packet is received, the receiver waits for delayed ACK timeout period before sending an ACK. This timeout period is usually 40 ms. This scenario may lead to spurious retransmissions if FG-RTO timers (as explained in the previous section) are deployed. The reason is that receiver waits for 40 ms before sending an ACK for the received packet and by that time, FG-RTO, which is in the order of a few microseconds, expires and forces the sender to retransmit the packet.

Thus, the delayed ACK timeout period either must be reduced to a few microseconds or must be completely disabled while using FG-RTOs to avoid such spurious retransmissions. This approach further enhances the TCP throughput in DCNs.

Advantages: It has been shown in [4] that reducing the delayed ACK timeout period to 200 μs while using FG-RTO achieves far better throughput than the throughput obtained when delayed ACKs are enabled. Moreover, completely disabling the delayed ACKs while using FG-RTO further improves the overall TCP throughput.

Shortcomings: The shortcomings of this approach are exactly similar to that of TCP with FG-RTO because this approach is an undesired side effect of the previous approach.

10.5.3 DCTCP [5]

Additive increase/multiplicative decrease (AIMD) is the cornerstone of TCP congestion control algorithms. When an acknowledgement (ACK) is received in AIMD phase, the congestion window (*cwnd*) is increased as shown in the following equation. This is known as additive increase phase of the AIMD algorithm:

$$cwnd = cwnd + \frac{1}{cwnd} \tag{10.1}$$

When congestion is detected either through *dupacks* or selective acknowledgement (SACK), *cwnd* is updated as shown in the following equation. This is known as multiplicative decrease phase of the AIMD algorithm:

$$cwnd = \frac{cwnd}{2} \tag{10.2}$$

Data center TCP (DCTCP) employs an efficient multiplicative decrease mechanism that reduces the *cwnd* based on the *amount of congestion* in the network rather than reducing it by half. DCTCP leverages explicit congestion notification (ECN) mechanism [13] to extract multibit feedback on the *amount of congestion* in the network from the single-bit stream of ECN marks. The next subsection describes the working of ECN in brief:

10.5.3.1 ECN

ECN [13] is one of the most popular congestion signaling mechanisms in communication networks. It is widely deployed in a large variety of operating systems at end hosts, modern Internet routers, and used by a variety of transport protocols. Moreover, it has been noticed that the use of ECN in the Internet has increased by threefolds in the last few years.

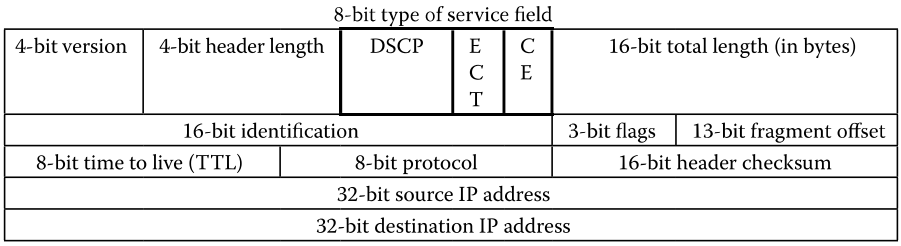

FIGURE 10.7
ECN bits in IP header.

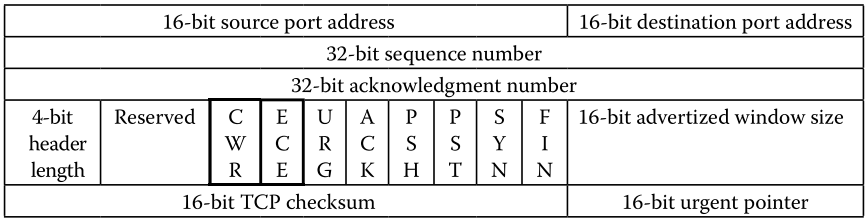

FIGURE 10.8
ECN bits in TCP header.

As shown in Figures 10.7 and 10.8, ECN uses two bits in the IP header, namely ECN-capable transport (ECT) and congestion experienced (CE), and two bits in the TCP header, namely congestion window reduced (CWR) and ECN echo (ECE), for signaling congestion to the end hosts. ECN is an industry standard, and its detailed mechanism is described in RFC 3168. Tables 10.5 and 10.6 show the ECN codepoints in the TCP header and the IP header, respectively, and Figure 10.9 shows in brief the steps involved in the working of ECN mechanism.

As described in RFC 3168, the sender and the receiver must negotiate the use of ECN during the three-way handshake (see Figure 10.10). If both are ECN capable, the sender marks every outgoing data packet with either ECT(1) codepoint or ECT(0) codepoint. This serves as an indication to the router that both sender and receiver are ECN capable. Whenever congestion builds up, the router marks the data packet by replacing ECT(1) or ECT(0) codepoint

TABLE 10.5
ECN Codepoints in the TCP Header

Codepoint	CWR Bit Value	ECE Bit Value
Non–ECN setup	0	0
ECN echo	0	1
CWR	1	0
ECN setup	1	1

TABLE 10.6
ECN Codepoints in the IP Header

Codepoint	ECT Bit Value	CE Bit Value
Non-ECT	0	0
ECT(1)	0	1
ECT(0)	1	0
CE	1	1

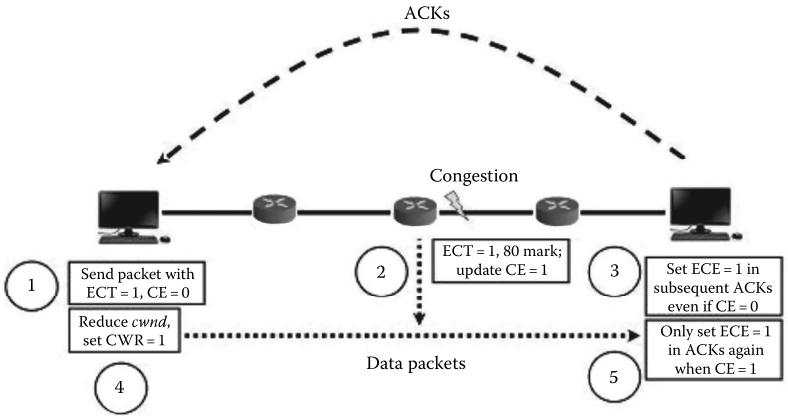

FIGURE 10.9
ECN.

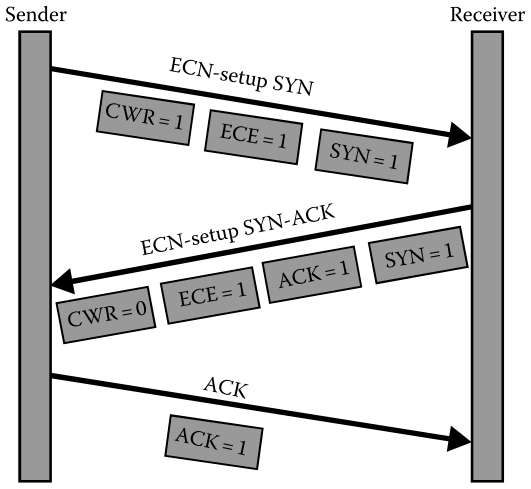

FIGURE 10.10
ECN negotiation.

by the CE codepoint. When the receiver receives a marked packet with CE codepoint, it infers congestion and hence marks *a series of* outgoing acknowledgments (ACKs) with ECE codepoint until the sender acknowledges with CWR codepoint (see Figure 10.9).

The major observation here is that, even if the router marks just one data packet, the receiver continues to mark ACKs with ECE until it receives confirmation from the sender (see Step 3 of Figure 10.9). This is to ensure the reliability of congestion notification, because even if the first marked ACK is lost, other marked ACKs would notify the sender about congestion. Note that this basic working of ECN aims to only notify the sender about congestion. It is not designed to provide the additional information about the *amount of congestion* to the sender.

At the receiver, counting the number of packets marked by the router provides fairly accurate information about the *amount of congestion* in the network. However, conveying this information to the sender by using ECN is a complex task. One of the possible ways is to enable the sender to count the number of marked ACKs it receives from the receiver. The limitation, however, is that even if router marks just one data packet, the receiver sends *a series of* marked ACKs. Hence, the number of marked ACKs counted by the sender would be much higher than the number of packets actually marked by the router. This would lead to incorrect estimation of the *amount of congestion* in the network.

To overcome this limitation, DCTCP modifies the basic mechanism of ECN. Unlike TCP receiver, which sends *a series of* marked ACKs, DCTCP receiver sends a marked ACK *only when* it receives a marked packet from the router, that is, it sets ECE codepoint in the outgoing ACK *only when* it receives a packet with CE codepoint. Thus, the DCTCP sender obtains an accurate number of packets marked by the router by simply counting the number of marked ACKs it receives. Note that this modification to the original ECN mechanism reduces the reliability because if a marked ACK is lost, sender remains unaware of the congestion and does not reduce the sending rate. However, since DCNs are privately controlled networks, the possibility that an ACK gets lost is negligible.

On receiving the congestion notification via ECN, the *cwnd* in DCTCP is reduced as shown in the following:

$$cwnd = cwnd \times \left(1 - \frac{\alpha}{2}\right) \tag{10.3}$$

where $\alpha (0 \leq \alpha \leq 1)$ is an estimate of the fraction of packets that are marked and is calculated as shown in (10.4). F in (10.4) is the fraction of packets that are marked in the previous *cwnd* and g $(0 < g < 1)$ is the exponential weighted moving average constant. Thus, when congestion is low (α *is near* 0), *cwnd* is

reduced slightly and when congestion is high (α *is near* 1), *cwnd* is reduced by half, just like the traditional TCP:

$$\alpha = (1 - g) \times \alpha + g \times F \tag{10.4}$$

The major goal of DCTCP algorithm is to achieve low latency (desirable for mice traffic), high throughput (desirable for elephant traffic), and high burst tolerance (to avoid packet losses due to incast). DCTCP achieves these goals by reacting to the *amount of congestion* rather than halving the *cwnd*. DCTCP uses a marking scheme at switches that sets the CE codepoint [13] of packets as soon as the buffer occupancy exceeds a fixed predetermined threshold, K (17% as mentioned in [14]). The DCTCP source reacts by reducing the window by a factor that depends on the fraction of marked packets: the larger the fraction, the bigger the decrease factor.

Advantages: DCTCP is a novel TCP variant that alleviates TCP incast, queue buildup, and buffer pressure problems in DCNs. It requires minor modifications to the original design of TCP and ECN to achieve these performance benefits. DCTCP employs a *proactive* behavior, that is, it tries to avoid packet loss. It has been shown in [5] that when DCTCP uses FG-RTO, it further reduces the impact of TCP incast and also improves the scalability of DCTCP. The stability, convergence, and fairness properties of DCTCP [14] make it a suitable solution for implementation in DCNs. Moreover, DCTCP is already implemented in the latest versions of Microsoft Windows server operating system.

Shortcomings: The performance of DCTCP falls back to that of TCP when the degree of incast increases beyond 35, that is, if there are more than 35 worker nodes sending data to the same aggregator, DCTCP fails to avoid incast and its performance falls back to that of the traditional TCP. However, authors show that dynamic buffer allocation at the switch and usage of FG-RTO can scale DCTCP's performance to handle up to 40 worker nodes in parallel.

Although DCTCP uses a simple queue management mechanism at the switch, it is ambiguous whether DCTCP can alleviate the problem of TCP outcast. Similarly, DCTCP does not address the problem of pseudocongestion effect in virtualized data centers. DCTCP utilizes minimum buffer space in the switches, which, in fact, is a desirable property to avoid TCP outcast. However, experimental studies are required to conclude whether DCTCP can mitigate the problems of TCP outcast and pseudocongestion effect.

10.5.4 ICTCP [9]

Like DCTCP, the main idea of incast congestion control for TCP (ICTCP) is to *avoid* packet losses due to congestion rather than to avoid from the packet losses. It is well known that a TCP sender can send a minimum of advertised

window (*rwnd*) and congestion window (*cwnd*) (i.e., min (*rwnd*, *cwnd*)). ICTCP leverages this property and efficiently varies the *rwnd* to avoid TCP incast. The major contributions of ICTCP are the following: (1) The available bandwidth is used as a quota to coordinate the *rwnd* increase of all connections, (2) per-flow congestion control is performed independently, and (3) *rwnd* is adjusted based on the ratio of difference between expected throughput and measured throughput over expected throughput. Moreover, live RTT is used for the throughput estimation.

Advantages: Unlike DCTCP, ICTCP does not require any modifications at the sender side (i.e., worker nodes) or network elements such as routers and switches. Instead, ICTCP requires modification only at the receiver side (i.e., an aggregator). This approach is adopted to retain the backward compatibility and make the algorithm general enough to handle the incast congestion in future high-bandwidth, low-latency networks.

Shortcomings: Although it has been shown in [9] that ICTCP achieves almost zero timeouts and high throughput, the scalability of ICTCP is a major concern, that is, the capability to handle incast congestion when there are extremely large number of worker nodes since ICTCP employs per-flow congestion control. Another limitation of ICTCP is that it assumes that both the sender and the receiver are under the same switch, which might not be always the case. Moreover, it is not known how much buffer space is utilized by ICTCP. Thus, it is difficult to conclude whether ICTCP can alleviate queue buildup, buffer pressure, and TCP outcast problems. Like DCTCP, ICTCP too does not address the problem of pseudocongestion effect in virtualized data centers.

10.5.5 IA-TCP [15]

Unlike DCTCP and ICTCP that use window-based congestion control, incast avoidance algorithm for TCP (IA-TCP) uses a rate-based congestion control algorithm to control the total number of packets injected in the network. The motivation behind selecting rate-based congestion control mechanism is that window-based congestion control mechanisms in DCNs have limitations in terms of scalability, that is, number of senders in parallel.

The main idea of IA-TCP is to limit the total number of outstanding data packets in the network so that it does not exceed the bandwidth-delay product (BDP). IA-TCP employs ACK regulation at the receiver and, like ICTCP, leverages the advertised window (*rwnd*) field of the TCP header to regulate the *cwnd* of every worker node. The minimum *rwnd* is set to 1 packet. However, if a large number of worker nodes send packets with respect to a minimum *rwnd* of 1 packet, the total number of outstanding packets in the network may exceed the link capacity. In such scenarios, IA-TCP adds delay, Δ, to the ACK packet to ensure that the aggregate data rate does not exceed the link capacity. Moreover, IA-TCP also uses delay, Δ, to avoid the synchronization among the worker nodes while sending the data.

Advantages: Like ICTCP, IA-TCP also requires modification only at the receiver side (i.e., an aggregator) and does not require any modifications at the sender or network elements. IA-TCP achieves high throughput and significantly improves the query completion time. Moreover, the scalability of IA-TCP is clearly demonstrated by configuring up to 96 worker nodes sending data in parallel.

Shortcomings: Similar to the problem of ICTCP, it is not clear how much buffer space is utilized by IA-TCP. Thus, experimental studies are required to conclude whether IA-TCP can mitigate queue buildup, buffer pressure, and TCP outcast problems. Like DCTCP and ICTCP, studies are required in virtualized data center environments to analyze the performance of IA-TCP with respect to the problem of pseudocongestion effect.

10.5.6 D²TCP [16]

Deadline-aware data center TCP (D²TCP) is a novel TCP-based transport protocol that is specifically designed to handle high burst situations. Unlike other TCP variants that are deadline agnostic, D²TCP is deadline aware. D²TCP uses a *distributed* and *reactive* approach for bandwidth allocation and employs a novel deadline-aware *congestion avoidance* algorithm that uses ECN feedback and deadlines to vary the sender's *cwnd* via a gamma-correction function [16].

D²TCP does not maintain per-flow information and, instead, inherits the distributed and reactive nature of TCP while adding deadline awareness to it. Similarly, D²TCP employs its congestion avoidance algorithm by adding deadline awareness to DCTCP. The main idea, thus, is that far-deadline flows back off aggressively and the near-deadline flows back off only a little or not at all.

Advantages: The novelty of D²TCP lies in the fact that it is deadline aware and reduces the fraction of missed deadlines up to 75% as compared to DCTCP. In addition, since it is designed upon DCTCP, it avoids TCP incast and queue buildup and has high burst tolerance.

Shortcomings: The shortcomings of D²TCP are exactly similar to those of DCTCP: scalability and whether it is robust against TCP outcast as well as pseudocongestion effect. However, since it is deadline aware, it would be interesting to analyze the robustness of D²TCP against the pseudocongestion effect in virtualized data centers.

10.5.7 TCP-FITDC [17]

TCP-FITDC is an adaptive delay-based mechanism to *prevent* the problem of TCP incast. Like D²TCP, TCP-FITDC is also a DCTCP-based TCP variant that benefits from the novel ideas of DCTCP. Apart from utilizing ECN as an indicator of network buffer occupancy and buffer overflow, TCP-FITDC also monitors changes in the queueing delay to estimate variations in the available bandwidth.

If there is no marked ACK received during the RTT, TCP-FITDC implies the queue length in the switch is below the marking threshold, and hence, TCP-FITDC increases the *cwnd* to improve the throughput. If marked ACKs are received during the RTT, *cwnd* is decreased to control the queue length. TCP-FITDC maintains two separate variables called rtt_1 and rtt_2 for unmarked ACKs and marked ACKs, respectively. By analyzing the difference between these two types of ACKs, TCP-FITDC gets more accurate estimate of the network conditions. The *cwnd* is then reasonably decreased to maintain low queue length.

Advantages: TCP-FITDC gets a better estimate of the network conditions by coupling the information received via ECN and the information obtained by monitoring the RTT. Thus, it has better scalability than DCTCP and scales up to 45 worker nodes in parallel. It avoids TCP incast and queue buildup and has a high burst tolerance because it is built upon DCTCP.

Shortcomings: The shortcomings of TCP-FITDC are similar to those of DCTCP, except that it improves the scalability of DCTCP. Unlike D²TCP, TCP-FITDC is deadline agnostic, and like all the aforementioned TCP variants, it does not address TCP outcast and pseudocongestion effect problems.

10.5.8 TDCTCP [18]

TDCTCP attempts to improvise the working of DCTCP (thus, it is DCTCP based) by making three modifications. First, unlike DCTCP, TDCTCP not only decreases but also increases the *cwnd* based on the *amount of congestion* in the network, that is, instead of increasing the *cwnd* as shown in (10.1), TDCTCP increases the *cwnd* as shown in (10.5). Thus, when the network is lightly loaded, the increment in *cwnd* is high, and vice versa:

$$cwnd = cwnd \times \left(1 + \frac{1}{1 + (\alpha/2)}\right) \tag{10.5}$$

Second, TDCTCP resets the value of α after every delayed ACK timeout. This is done to ensure that α does not carry the stale information about the network conditions, because if the stale value of α is high, it restricts the *cwnd* increment and thereby degrades the overall throughput. Lastly, TDCTCP employs an efficient approach to dynamically calculate the delayed ACK timeout with a goal to achieve better fairness.

Advantages: TDCTCP achieves 26%–37% better throughput and 15%–20% better fairness than DCTCP in a wide variety of scenarios ranging from single bottleneck topologies to multibottleneck topologies and varying buffer sizes. Moreover, it achieves better throughput and fairness even at very low values of *K*, that is, the ECN marking threshold at the switch. However, there is a slight increase in the delay and queue length while using TDCTCP as compared to DCTCP.

Shortcomings: Although TDCTCP improves throughput and fairness, it does not address the scalability challenges faced by DCTCP. Like most of the other TCP variants discussed, TDCTCP too is deadline agnostic and does not alleviate the problems of TCP outcast and pseudocongestion effect.

10.5.9 TCP with Guarantee Important Packets (GIP) [19]

TCP with GIP mainly aims to improve the network performance in terms of goodput by minimizing the total number of timeouts. Timeouts lead to a dramatic degradation in the network performance and affect the user perceived delay. The authors of TCP with GIP focus on avoiding mainly two types of timeouts in the network: (1) the timeouts due to the loss of full window of packets, full window loss timeouts (FLoss-TOs), and (2) the timeouts due to the lack of ACKs, lack of ACKs timeouts (LAck-TOs).

FLoss-TOs generally occur when the total data sent by all the worker nodes exceed the available bandwidth in the network, and thus, a few unlucky flows end up losing all the packets of the window. On the other hand, LAck-TOs mainly occur when the transmission is *barrier-synchronized transmission*. In such transmissions, the aggregator will not request the worker nodes to transmit the next stripe units until all the worker nodes finish sending their current ones. If a few packets get dropped at the end of the stripe unit, they cannot be recovered until the RTO fires because there may not be sufficient *dupacks* to trigger fast retransmit.

TCP with GIP introduces *flags* in the interface between the application layer and the transport layer. These *flags* indicate whether the running application follows many-to-one communication pattern or not. If the running application follows such a communication pattern, TCP with GIP redundantly transmits the last packet of the stripe unit at most three times and each worker node decreases its initial *cwnd* at the head of the stripe unit. On the other hand, if the running application does not follow many-to-one communication pattern, TCP with GIP works like a standard TCP.

Advantages: TCP with GIP achieves almost zero timeouts and higher goodput in a wide variety of scenarios including with and without the background UDP traffic. Moreover, the scalability of TCP with GIP is much more than any other TCP variant discussed earlier, that is, it scales well up to 150 worker nodes in parallel.

Shortcomings: TCP with GIP does not address the queue occupancy problem resulting from the presence of elephant traffic. As a result, the queue buildup, buffer pressure, and TCP outcast problems remain unsolved because all these problems arise due to the lack of the buffer space in the switches. Although timeouts are eliminated by TCP with GIP, flows may miss the specified deadlines because of queueing delay. Moreover, the hypervisor scheduling latency is not taken into consideration, and thus, the problem

of pseudocongestion effect also remains open. Note that high latencies introduced by hypervisor scheduling algorithm may also prevent flows from meeting the specified deadlines.

10.5.10 PVTCP [7]

Paravirtualized TCP (PVTCP) proposes an efficient solution to the problem of pseudocongestion effect. This approach does not require any changes to be done in the hypervisor. Instead, the basic working of TCP is modified to accept the latencies introduced by the hypervisor scheduler. An efficient approach is suggested to capture the *actual* picture of every packet transmission involving the hypervisor-introduced latencies and then determine RTO more accurately to filter out pseudocongestion effect.

Whenever the hypervisor introduces scheduling latency, sudden spikes can be observed during the regular measurements of RTT. PVTCP detects these sudden spikes and filters out the negative impact of these spikes by proper RTT measurement and RTO management. While calculating average RTT, PVTCP ignores the measurement of a particular RTT if a spike is observed in that RTT.

Advantages: PVTCP solves the problem of pseudocongestion effect without requiring any changes in the hypervisor. By detecting the unusual spikes, accurately measuring RTT, and properly managing RTO, PVTCP enhances the performance of virtualized data centers.

Shortcomings: The scalability of PVTCP is ambiguous, and thus, whether it can solve TCP incast effectively or not is unclear. The queue occupancy while using PVTCP is not taken into consideration, which may further lead to problems such as queue buildup, buffer pressure, TCP outcast, and missed deadlines.

10.5.11 Summary: TCP Enhancements for DCNs

Table 10.7 summarizes the comparative study of TCP variants proposed for DCNs. Apart from the novelty of the proposed TCP variant, the table also highlights the deployment complexity of each protocol. The protocols that require modifications in the sender, receiver, and switch are considered as hard to deploy. The ones that require modification only at the sender or receiver are considered as easy to deploy. DCNs, however, are privately controlled and managed networks, and thus, the former ones may also be treated as easy to deploy.

Apart from the aforementioned parameters, the summary also includes which problems among TCP incast, TCP outcast, queue buildup, buffer pressure, and pseudocongestion effect are alleviated by each TCP variant. The details regarding the tools used/approach of implementation adopted by the authors are also listed.

TABLE 10.7

Summary of TCP Variants Proposed for DCNs

TCP Variants Proposed for DCNs	Modifies Sender	Modifies Receiver	Modifies Switch	Solves TCP Incast	Solves TCP Outcast	Solves Queue Buildup	Solves Buffer Pressure	Is Deadline Aware?	Detects Pseudocongestion	Implementation
TCP with FG-RTO	√	x	x	√	x	X	x	x	x	Testbed and ns-2
TCP with FG-RTO + delayed ACKs disabled	√	x	x	√	x	X	x	x	x	Testbed and ns-2
DCTCP	√	√	√	√	x	√	√	x	x	Testbed and ns-2
ICTCP	x	√	x	√	x	x	X	x	x	Testbed
IA-TCP	x	√	x	√	x	x	x	x	x	ns-2
D²TCP	√	√	√	√	x	√	√	√	x	Testbed and ns-3
TCP-FITDC	√	√	√	√	x	√	√	x	x	Modeling and ns-2
TDCTCP	√	√	√	√	x	√	√	x	x	OMNeT++
TCP with GIP	x	√	x	√	x	x	x	x	x	Testbed and ns-2
PVTCP	√	√	x	√	x	x	x	x	√	Testbed

Although several modifications have been proposed to the original design of TCP, there is an acute need to further optimize the performance of TCP in DCNs. A few open issues are listed in the following:

1. Except D^{2}TCP, all other TCP variants are deadline agnostic. Meeting deadlines is the most important requirement in DCNs. Missed deadlines may lead to violations of SLAs and, thus, incur high cost to the organization.
2. Most of the data centers today employ virtualization for efficient resource utilization. Hypervisor scheduling latency ranges from microseconds to hundreds of milliseconds and, hence, may hinder in successful completion of flows within the specified deadline. While making modifications to hypervisors is one viable solution, designing an efficient TCP that is deadline aware and automatically tolerates hypervisor scheduling latency is a preferred solution.
3. A convincing solution to TCP outcast problem is unavailable. An optimal solution to overcome TCP outcast must ensure minimal buffer occupancy at the switch. Since RED is implemented in most of the modern switches, it can be used to control the buffer occupancy. The parameter sensitivity of RED, however, poses further challenges and complicates the problem.

10.6 Summary

Data centers in the present scenario house a plethora of Internet applications. These applications are diverse in nature and have various performance requirements. Understanding the complete architecture of a data center from the point of its physical as well as networking infrastructure is crucial to the success of cloud computing.

Although several issues related to networking still exist in data centers, one of the most crucial ones is to meet the diverse requirements of various traffic types that coexist in a data center environment. Majority of the traffic uses many-to-one communication pattern to gain performance efficiency. TCP, which has been a mature transport protocol of Internet since the past several decades, suffers from performance impairments such as TCP incast, TCP outcast, queue buildup, buffer pressure, and pseudocongestion effect in DCNs. We described each of the aforementioned impairment in brief along with the causes and possible approaches to mitigate them. A comparative study of TCP variants that have been specifically designed for DCNs is presented, while describing the advantages and shortcomings of each.

Review Points

- Data centers are mainly classified based on the uptime they provide to the cloud user (see Section 10.1).
- Around 15% of the cost in data centers goes in networking (see Section 10.3).
- Different traffic types with diverse performance requirement coexist in DCNs (see Section 10.2.6).
- Traditional TCP/IP does not provide optimal network performance in DCNs (see Section 10.4).
- Although a few new TCP algorithms have been designed to enhance the network performance in data centers, a lot remains to be done (see Section 10.5).

Review Questions

1. What are the different ways to classify data centers?
2. Explain the different types of traffic in DCNs. Mention their performance requirements and provide an example for each type of traffic.
3. How is queue buildup different from buffer pressure?
4. Describe at least two approaches to solve the problem of pseudocongestion in virtualized data centers.
5. Why traditional ECN mechanism cannot be used in DCTCP?

References

1. Zhang, J., F. Ren, and C. Lin. Survey on transport control in data center networks. *IEEE Network* 27(4): 22–26, 2013.
2. Greenberg, A., J. Hamilton, D. A. Maltz, and P. Patel. The cost of a cloud: Research problems in data center networks. *SIGCOMM Computer Communication Review* 39(1): 68–73, December 2008. Available [Online]: http://doi.acm.org/10.1145/1496091.1496103.
3. Kant, K. Data center evolution. *Computer Networks* 53(17): 2939–2965, December 2009. Available [Online]: http://dx.doi.org/10.1016/j.comnet.2009.10.004.

4. Vasudevan, V., A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, and B. Mueller. Safe and effective fine-grained TCP retransmissions for datacenter communication. *SIGCOMM Computer Communications Review* 39(4): 303–314, August 2009. Available [Online]: http://doi.acm.org/10.1145/1594977.1592604.
5. Alizadeh, M., A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. Data center TCP (DCTCP). *SIGCOMM Computer Communications Review* 40(4): 63–74, August 2010. Available [Online]: http://doi.acm.org/10.1145/1851275.1851192.
6. Prakash, P., A. Dixit, Y. C. Hu, and R. Kompella. The TCP outcast problem: Exposing unfairness in data center networks. *Proceedings of the Ninth USENIX Conference on Networked Systems Design and Implementation*, ser. *NSDI'12*. USENIX Association, Berkeley, CA, 2012, pp. 30–30. Available [Online]: http://dl.acm.org/citation.cfm?id=2228298.2228339.
7. Cheng, L., C.-L. Wang, and F. C. M. Lau. PVTCP: Towards practical and effective congestion control in virtualized datacenters. *21st IEEE International Conference on Network Protocols*, ser. *ICNP 2013*. IEEE, 2013.
8. Chen, Y., R. Griffith, J. Liu, R. H. Katz, and A. D. Joseph. Understanding TCP incast throughput collapse in datacenter networks. *Proceedings of the First ACM Workshop on Research on Enterprise Networking*, ser. *WREN'09*. ACM, New York, 2009, pp. 73–82. Available [Online]: http://doi.acm.org/10.1145/1592681.1592693.
9. Wu, H., Z. Feng, C. Guo, and Y. Zhang. ICTCP: Incast congestion control for TCP in data center networks. *Proceedings of the Sixth International Conference*, ser. *Co-NEXT'10*. ACM, New York, 2010, pp. 13:1–13:12. Available [Online]: http://doi.acm.org/10.1145/1921168.1921186.
10. Kurose, J. F. and K. W. Ross. *Computer Networking: A Top Down Approach*, 6th edn. Addison-Wesley, 2012.
11. Ren, Y., Y. Zhao, P. Liu, K. Dou, and J. Li. A survey on TCP incast in data center networks. *International Journal of Communication Systems* pp. n/a–n/a, 2012. Available [Online]: http://dx.doi.org/10.1002/dac.2402.
12. Floyd, S. and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking* 1: 397–413, August 1993. Available [Online]: http://dx.doi.org/10.1109/90.251892.
13. Ramakrishnan, K. K. and S. Floyd. The addition of explicit congestion notification (ECN) to IP, 2001, rFC 3168.
14. Alizadeh, M., A. Javanmard, and B. Prabhakar. Analysis of DCTCP: Stability, convergence and fairness. *Proceedings of the ACM SIGMETRICS, Joint International Conference on Measurement and Modeling of Computer Systems*, ser. *SIGMETRICS'11*. ACM, New York, 2011, pp. 73–84. Available [Online]: http://doi.acm.org/10.1145/1993744.1993753.
15. Hwang, J., J. Yoo, and N. Choi. IA-TCP: A rate based incast-avoidance algorithm for TCP in data center networks. *IEEE ICC 2012*, Ottawa, Canada, 2012.
16. Vamanan, B., J. Hasan, and T. Vijaykumar. Deadline-aware datacenter TCP (D2TCP). *SIGCOMM Computer Communications Review* 42(4): 115–126, August 2012. Available [Online]: http://doi.acm.org/10.1145/2377677.2377709.

17. Wen, J., W. Zhao, J. Zhang, and J. Wang. TCP-FITDC: An adaptive approach to TCP incast avoidance for data center applications. *Proceedings of the 2013 International Conference on Computing, Networking and Communications (ICNC)*, ser. *ICNC'13*. IEEE Computer Society, Washington, DC, 2013, pp. 1048–1052. Available [Online]: http://dx.doi.org/10.1109/ICCNC.2013.6504236.
18. Das, T. and K. M. Sivalingam. TCP improvements for data center networks. *2013 Fifth International Conference on Communication Systems and Networks (COMSNETS)*. IEEE, 2013, pp. 1–10.
19. Zhang, J., F. Ren, L. Tang, and C. Lin. Taming TCP incast throughput collapse in data center networks. *21st IEEE International Conference on Network Protocols*, ser. *ICNP 2013*. IEEE, 2013.