Survey on Transport Control in Data Center Networks

Jiao Zhang, Fengyuan Ren, and Chuang Lin, Tsinghua University

Abstract

Traditional fair bandwidth sharing by leveraging AIMD-based congestion control mechanisms faces great challenges in data center networks. Much work has been done to solve one of the various challenges. However, no single transport layer protocol can solve all of them. In this article, we focus on the transport layer in data centers, and present a comprehensive survey of existing problems and their current solutions. We hope that this article can help readers quickly understand the causes of each problem and learn about current research progress, so as to help them make new contributions in this field.

Data centers are pools that provide large volumes of compute and storage resources to support today's Internet services. They can be classified into two main types according to their functions. One type aims to provide online services to customers, such as the Google search engine, Facebook, and Yahoo!. The other type offers resources to users on a pay-as-you-go model, such as Amazon EC2 and Microsoft Azure.

Data centers have many unique features.

- Multiple paths. To support intensive communication among servers and provide strong fault tolerance, multiple paths between any two servers are provided in many recently designed data center infrastructures.
- Small propagation delay. Data centers are commonly located in close proximity; the propagation round-trip delay is on the magnitude of microseconds instead of the milliseconds in traditional Internet.
- Mixture of long and short flows. The traffic in data centers consists of a large proportion of latency-sensitive mice flows and a small proportion of elephant flows. User experience of online services primarily depends on the response time of short flows. Thus, it is important to reduce the latency for short flows while keeping high throughput for long flows.
- Special communication pattern. Many-to-one or many-to-many communication patterns widely exist in computing systems (MapReduce, Dryad, Spark), distributed file storage systems, and web applications that are designed based on partition/aggregation workflow.
- Virtualization. To flexibly allocate resources, isolate services, reduce the cost of system maintenance, and so on, virtualization is largely used in cloud data centers.
- Unified fabric. To improve asset utilization and reduce capital expenses, Ethernet is being enhanced as a unified data center fabric that supports IP communication traffic, storage data, and high-performance computing traffic. To support the three types of traffic, some new technologies are developed, such as quantized congestion notification (QCN) and priority-based flow control (PFC).

These features pose lots of challenges to both types of data centers; for example, the TCP incast problem caused by the many-to-one traffic pattern, large latency of online queries, TCP performance deterioration in virtualized data centers, and malicious users in multi-tenant data centers. Much work has been done to address these challenges. In this survey, we mainly focus on the problems related to the transport layer. Regarding each problem, we introduce the background and causes, present the existing solutions, and discuss the challenges and opportunities.

TCP Incast

Background and Causes

The incast communication pattern can be described as follows: many senders transmit data to a single receiver in units of data blocks, and no sender can transmit the next data block until all the senders finish transmitting the current data blocks. This kind of transmission is called *barrier synchronized transmission* in [1].

The TCP incast problem widely exists in today's data centers:

- In distributed storage systems. Numerous data are stored in many distributed nodes, such as BigTable or HBase. When a client retrieves data, parallel access to some of these distributed nodes is needed.
- In data-intensive scalable computing systems, such as MapReduce, Dryad, and Spark. These systems quickly deal with large amounts of data by parallel processing across many servers. Thus, all-to-all or many-to-one transmissions are needed to transfer data between nodes. Data-intensive applications have been increasing in various fields, including web-search, e-commerce, and social networks.
- In partition/aggregation workflows. In most large-scale web applications, every requested task is broken into small pieces and assigned to the workers in the lower layer. Then the responses from multiple workers are transmitted to the aggregator and generated into the final search result. This partition/aggregation design naturally incurs a many-to-one communication pattern.

In the incast communication pattern, if the block size is large enough, TCP works normally without throughput collapse [2]. The main difference between the barrier synchro-

0890-8044/13/$25.00 © 2013 IEEE

nized transmission and normal transmission is that the senders in the former pattern have to wait for each other at the boundaries of each data block. Such synchronization incurs two kinds of timeouts at the boundaries of data blocks that dramatically decrease throughput. This does not happen in normal TCP flows [3].

Existing Solutions

The TCP incast problem is attracting much attention due to its wide existence in cloud applications and dramatic performance deterioration. Three kinds of solutions have been proposed to address the problem.

Revising TCP — V. Vasudevan *et al.* proposed reducing the minimum retransmission timeout period (RTOmin) of TCP from 200 ms to about 2 ms to decrease the link idle time caused by lots of timeouts [4]. By leveraging the fact that there is only one receiver in the incast communication pattern, ICTCP [1] controls the sender rates through setting proper TCP receive window(rwnd) and thus reduces the number of timeouts.

Replacing TCP — To solve the TCP incast problem, Facebook replaces TCP with UDP and shifts the burden of reliability to the application layer. Some other new transport protocols, which mainly focus on reducing latency [5, 6], can also alleviate the performance deterioration of the TCP incast problem.

Solving TCP Incast at Other Layers — A. Phanishayee *et al.* pointed out that enabling Ethernet flow control would be helpful to improve the goodput in the incast communication pattern. Y. Zhang *et al.* investigated the performance of QCN in the incast communication pattern and found that the performance is not good due to the unfairness among different flows. Thus, fair QCN (FQCN) is proposed to improve the fairness of multiple flows sharing one bottleneck link [7].

Challenges and Opportunities

Challenges — The TCP incast problem is related to many factors, including network parameters, such as the switch buffer size, link capacity, and number of senders, as well as the barrier synchronized transmission imposed by the application layer. Traditional congestion control at the transport layer only needs to fairly transfer more packets for each flow. However, barrier synchronized transmission in the incast communication pattern requires that the congestion control mechanisms take the interactions of all connections into consideration.

Opportunities — Data centers are generally managed by a single organization, which provides the opportunity to design a novel transport protocol from a clean slate to solve the problems caused by TCP. Since the throughput collapse in the TCP incast problem is caused by a large number of timeouts, one possible direction is to design a lossless transport protocol to radically eliminate the timeouts. The lossless congestion control mechanisms designed for other networks, such as asynchronous transfer mode (ATM) and infiniband, could be used for reference. The other direction is to set a proper delay value for each flow (e.g., by the receiver) and design delay-guaranteed transport protocols to avoid the throughput collapse. There is some literature that aims to guarantee the flow latency. If the synchronized incast flows have the same latency, they could be completed near the same time with protocols that guarantee flow latency.

Latency

Background and Causes

Low latency is a critical requirement of companies that provide online services. User experience is negatively impacted by large latency. Amazon found that every 100 ms of latency costs them 1 percent in sales [8]. Google found an extra 0.5 s in search page generation time dropped traffic by 20 percent [9]. Also, larger latency may severely affect the business. A broker could lose $4 million in revenue per millisecond if their electronic trading platform is 5 ms behind the competition [10].

One important factor that causes large latency in today's web applications is service dependent latency. After Facebook receives an HTTP request for a web page, the application server has to make an average of 130 internal requests inside the Facebook site as part of generating the HTML for the page. The internal requests are sequentially dependent, which causes a large cumulative latency. Amazon also reports that it needs 100–200 requests to generate HTML for each page [11]. Since later requests need the results generated by the earlier requests, the communication between servers to transfer data will cost a lot of time. Therefore, reducing the communication latency between servers plays an important role in improving the speed of web applications.

Discussion of the latency caused by the service framework is beyond the scope of this article. From the network view, the main cause of large latency is queue buildup. The propagation round-trip delay in data centers only takes dozens of microseconds. However, queue buildup can lead to dozens of milliseconds of latency [6]. The widely used TCP protocol only provides fair bandwidth sharing; obviously, it cannot satisfy the delay requirements of flows.

Existing Solutions

Large queue buildup in data center networks causes short flows to suffer large latency. Therefore, DCTCP [6] employs the early congestion notification (ECN) mechanism to control the queue length within a small value in order to reduce the latency of short flows. However, this method only decreases the latency of short flows; it cannot guarantee specific delay requirements and does not provide differentiated services for flows with different delay requirements.

Afterward, several mechanisms have been proposed to guarantee delay requirements. There are two main classes. One class attempts to compute the flow rate according to the delay information of the flows [5, 12]. D^2TCP extends the window evolution function of DCTCP. The flows with smaller remaining delay will obtain higher rates. In D^3, the end hosts compute the desired rate for each delay-sensitive flow according to the flow size and deadline, then convey the computed rates to the switches. The switches allocate the output link capacity to each flow based on the collected rate values and the number of flows. The flows without deadline fairly share the spare bandwidth. The senders of flows need to send their desired rates to the switches periodically, which brings some overhead.

The other class reduces the latency for delay-sensitive flows through priority-based scheduling at the switches [13–15]. PDQ enables flow preemption to approximate shortest job first (SJF) and earlier deadline first (EDF) policies. PDQ borrows ideas from centralized scheduling disciplines and implements them in a fully distributed manner, making it scalable to today's data centers [13]. DeTail is a cross-layer mechanism, with congestion-aware packet-level routing and the priority queue mechanism in the link layer. PFC reduces the latency of flows [14]. In pFabric, the end hosts put a priority

in each packet header. The switch always sends the packet with the highest priority and drops the packet with the lowest priority [15].

Challenges and Opportunities

Challenges — User experience of online data-intensive services in data centers largely depends on latency. Thus, service providers have to focus not only on the average flow completion time, but also on the tail of the latency distribution, such as the 95th or 99th latency percentile of the distribution [14]. Guaranteeing the tail latency is quite challenging due to the varying computing demands of different queries. Some existing techniques, such as SJF scheduling, focus on reducing the average flow completion time instead of reducing the tail of latency.

Opportunities — First, data centers provide more abundant bandwidth in the form of multiple paths. Making full use of the redundant bandwidth will be helpful to reduce the flow latency.

Second, converged enhanced Ethernet (CEE) seeks to become a unified fabric of LAN, storage area networking (SAN), and high-performance computing (HPC). Some new link layer protocols with enhanced functions have been designed. It is possible to employ them to reduce the latency.

Third, current methods of reducing or guaranteeing latency generally need the flow size in advance. However, in practice, the size of a flow might not be available at the establishment phase. Therefore, designing a mechanism of reducing average flow completion time without knowing the flows sizes at the start of a connection is a possible direction.

Finally, online data-intensive services are a relatively new kind of workload. Each response is computed over huge amounts of data across many servers. Thus, the response time depends on many factors. Besides reducing the transmission delay, we can also reduce delay in terms of decreasing data access time, such as replacing disks with RAM [16], or designing a better service model to reduce the number of steps in generating a response.

Transport Problems in Virtualized Data Centers

Background and Causes

Virtualization is a key technology which helps data centers provide cloud services, such as software as a service (SAAS), platform as a service (PAAS), and infrastructure as a service (IAAS). The employment of virtual machine (VM) consolidation offers better server utilization, service isolation, and lower system maintenance cost. However, virtualization also changes the environment where protocols run. One problem attracting a lot of attention is the impact of virtualization on transport layer protocols. The measurement study of the Amazon Elastic Cloud Computing (EC2) data center [17] shows that virtualization dramatically deteriorates the performance of transport layer protocols. The throughput of both TCP and UDP becomes unstable. The end-to-end delay of packets is large even if the network load is light.

The essential reason for the unstable throughput and large delay is the large scheduling latency of hypervisors. In data centers, the propagation round-trip delay is within 1 ms. However, as more VMs are created in the same host machine, the waiting time for a VM to be scheduled by the hypervisor to access the core/CPU increases. Such scheduling latency can be as high as tens or hundreds of milliseconds [18], which is larger than the propagation delay. Thus, the last hop between the hypervisor and VMs becomes the bottleneck of the end-to-end transmission.

Existing Solutions

To mitigate the negative effect of the large scheduling latency, vSnoop and vFlood are proposed to solve the problem at the receiver and sender side of TCP, respectively. In vSnoop [18], a module is placed within the driver domain to generate acknowledgments (ACKs) on behalf of the receiving VM. Thus, the sender can get the feedback more quickly and thus ramp up faster. In vFlood [19], the congestion control functionality of TCP is offloaded to the driver domain. The driver domain handles congestion control on behalf of the VMs. The VMs will flood packets to the driver domain. However, when the number of connections on the same host increases, a larger buffer is required in the driver domain to accommodate the TCP packets from all the VMs located on the same physical host.

Challenges and Opportunities

Challenges — The virtual switch moves networking into the server realm. Traditionally, the network and server are studied by different groups. However, solving the network problem in virtualized data centers requires detailed knowledge of both the server and network, which is quite challenging.

Besides, the large scheduling latency impacts the applicability of some protocols designed for data centers. For example, pFabric [15] leverages the fact in data centers that the product of bandwidth and round-trip time is quite small, and designs a prioritized scheduling mechanism to satisfy the delay requirements of flows. However, it possibly does not apply to virtualized data centers with quite large RTT unless the scheduling latency of the hypervisor can be eliminated or largely reduced.

Opportunities — The maximum throughput of a flow is generally clamped by the bottleneck switch along the routing paths. Hypervisors at end hosts can be treated as a virtualized switch. Although it is hard for network providers to improve the performance of a hardware switch, it is possible to design mechanisms to improve the switching speed of the software switch. There are two possible directions. One is reducing the scheduling latency as vSnoop and vFlood do. The other one is eliminating the scheduling latency, that is, the packets do not pass through hypervisor, but are directly transmitted to VMs.

Bandwidth Sharing in Multi-Tenant Data Centers

This section focuses on the performance of one flow in virtualized data centers. Here we discuss the problem of allocating bandwidth to different tenants.

Background and Motivation

The technology of VM means that cloud data center providers can easily allocate CPU and memory to multiple tenants. However, bandwidth sharing is still a challenging problem. How should VMs share the network bandwidth? Does traditional fair sharing still work? Or should the bandwidth be allocated according to payment or other metrics? If we know the bandwidth requirements of different tenants or VMs, how can they be guaranteed? In eyeQ [20], the authors point out that the tenants would like to have predictable performance, as if the network allocated to them were dedicated. Clearly, the fair sharing bandwidth allocation of AIMD could not satisfy this requirement. Besides, TCP does little to isolate tenants from one another. Malicious applications can consume

ants from one another. Malicious applications can consume most of the network capacity. Since the number of tenants is quite large, and their bandwidth requirements could vary widely, it is not practical to use the existing router mechanisms, such as weighted fair queuing or reservation, to allocate bandwidth.

Existing Solutions

The bandwidth sharing mechanisms in multi-tenant data centers can mainly be classified into two categories. One is per tenant sharing. The other one is per VM sharing.

SeaWall [21] aims to provide performance isolation in multi-tenant data centers. Each entity, such as a VM, process, or collection of port numbers, that can be any traffic source confined to a single node is assigned a weight. The hypervisor allocates bandwidth for them according to the values of their weights. It is work-conserving since if an entity takes less bandwidth than it is allocated, the remaining bandwidth is allocated to other entities. However, how to automatically assign a proper weight value to each entity or change the weight values of current entities after new entities come is still a challenging problem, especially when the number of entities is quite large.

EyeQ [20] is motivated by the output queued switch, which eliminates the congestion by some scheduling mechanisms. It treats the whole network as a giant switch and conducts congestion control at the edge switches. EyeQ provides a minimum bandwidth guarantee to VMs by trading off a small fraction of the access link bandwidth. Therefore, eyeQ can avoid malicious tenants aggressively grabbing more bandwidth. However, it cannot eliminate in-network congestion.

FairCloud [22] first proposes a set of properties that the network sharing mechanism should have, and identifies a key trade-off between the ability to share congested links in proportion to payment and the ability to provide minimal bandwidth guarantees to VMs. Then it presents three allocation policies that provide minimum bandwidth guarantees and achieve better proportionality than existing solutions do.

Challenges and Opportunities

Challenges — The novel problem of allocating bandwidth to multiple tenants rises with the development of public cloud data centers and the pay-as-you-go model. Therefore, how to properly and clearly define the problem is quite critical. However, the definition is related to the requirements of customers and the profit of the service providers, which makes the problem definition quite challenging.

Opportunities — The first opportunity of this problem is to design new principles of sharing bandwidth among tenants or VMs. How bandwidth is shared directly impacts the tenants' quality of experience as well as the bandwidth utilization of cloud data centers. The principle could be related to the service model. For example, if differentiated services are provided to customers, it is necessary to provide prioritized bandwidth allocation for different kinds of tenants.

Once the principles are clearly defined, the corresponding solutions can be devised from different perspectives. Assigning proper global weight values is the first important step in providing different bandwidth values for tenants. The weight values directly determine the bandwidth allocation results. However, each entity does not know the weight values of the other entities. Then some local information could be used to determine a global effective weight value. For example, one simple method is using the bandwidth requirement as the weight value.

In terms of allocating bandwidth according to the weight values, switch mechanisms or congestion control mechanisms at the end host could be employed. There are some existing switch mechanism that support weighted fair share, such as Weighted Fair Queueing (WFQ) and Worst-Case Fair Weighted Fair Queueing (WF2Q). Since modifying switch functions incurs large cost, it is more practical to investigate whether the technologies supported by current switches could be directly used. At the end hosts, only a little work has been done to share bandwidth among entities. The typical work, SeaWall [21], designed a hypervisor-based congestion control mechanism to support weighted bandwidth sharing among flows. FairCloud [22] also states that they will implement their allocation policies using Seawall's mechanism. However, SeaWall does not provide performance predictability [23]. It is a good opportunity to design more effective congestion control mechanisms that support weighted bandwidth sharing.

Underutilization of Redundant Bandwidth

Background and Existing Solutions

Widespread use of distributed applications in data centers makes the aggregation layer and core layer of data center infrastructures often become bottlenecks. To address this problem, a number of new data center topologies with multiple paths are proposed. Flow-level random routing is a typical algorithm to utilize the multiple paths. However, since the flow size varies largely, flow-level random routing fails to balance load well. Possibly paths with several long flows become hotspots, while other paths are still idle.

MPTCP [24] takes advantage of multiple parallel paths between a pair of end nodes routinely found in data center environments. One flow is split into several subflows among the parallel paths. The rate of each subflow is dynamically adjusted according to the congestion information of the path. MPTCP provides higher fault tolerance ability since one flow will not terminate even if a path is broken.

Challenges and Opportunities

MPTCP will possibly bring out-of-order packets since the packets of a flow are transmitted along different paths. DeTail [14] points out that out-of-order packets are not a problem: the end hosts can use large buffer to accommodate them. However, if the number of subflows is large and the link capacity is high, the end hosts may not have enough buffer to accommodate all of the out-of-order packets. Thus, how to utilize the redundant bandwidth while avoiding out-of-order packets is challenging, and is also a possible future research direction.

Like TCP, MPTCP still provides fair bandwidth allocation. However, the applications in data centers may have specific performance requirements, such as small latency and/or guaranteed bandwidth. Thus, how to extend MPTCP to provide differentiated services is another possible future research direction.

TCP in CEE

Background

It is economic to design a unified data center network infrastructure for communication traffic, data storage, and high-performance computing data. Due to the widespread and easy management features of Ethernet, CEE becomes a potential candidate for the unified infrastructure. Two key features of CEE are 802.1Qbb PFC and 802.1Qau QCN. PFC provides priority-based flow control, while QCN provides congestion control in the link layer. However, how does TCP perform over CEE networks? Can TCP cooperate well with the link layer protocol? D. Crisan *et al.* studied the performance of

TCP over CEE [25]. They chose three data center workloads and three TCP variants, and conducted both simulations and experiments. The results show that PFC benefits TCP, while TCP performance over QCN highly depends on the type of workload and the communication pattern.

Opportunities

The mechanims provided in CEE have been employed to solve some problems faced by TCP. For example, QCN is modified to fairly allocate bandwidth to each flow and thus solve the TCP incast problem [7]. DeTail makes use of the PFC mechanism to give priority to delay-sensitive flows and thus satisfy their latency requirement. Since the link layer provides new functions, it is desirable to make full use of them to solve the existing problems and provide better services.

Besides, the cooperation of TCP and the congestion control mechanisms in the link layer is a very important issue in CEE. However, only a little work has been done on it. There is still much space to cover.

Conclusion

In this work, we present a survey of six problems related to the transport layer in data center networks, including throughput collapse in the incast communication pattern, large latency suffered by flows, abnormal performance in virtualized data centers, bandwidth sharing in multi-tenant data centers, underutilization of redundant bandwidth, and cooperation of TCP and link layer mechanisms in CEE. We describe why each problem happens, review current solutions, and discuss the challenges and opportunities, hoping the article can shed light on the research in this field.

Acknowledgment

The authors gratefully acknowledge the anonymous reviewers for their constructive comments and Hongkun Yang for polishing the article. This work is supported in part by the National Natural Science Foundation of China (NSFC) under Grant No. 61225011, and the National Basic Research Program of China (973 Program) under Grants No. 2012CB315803 and 2009CB320504.

References

[1] H. Wu *et al.*, "ICTCP: Incast Congestion Control for TCP in Data Center Networks," *ACM CoNEXT*, 2010.
[2] A. Phanishayee *et al.*, "Measurement and Analysis of TCP Throughput Collapse in Cluster-based Storage Systems," *USENIX FAST*, 2008.
[3] J. Zhang, F. Ren, and C. Lin, "Modeling and Understanding TCP Incast in Data Center Networks," *IEEE INFOCOM*, 2011, pp. 1377–85.
[4] V. Vasudevan *et al.*, "Safe and Effective Finegrained TCP Retransmissions for Datacenter Communication," *ACM SIGCOMM*, 2009, pp. 303–14.
[5] C. Wilson and T. Karagiannis, "Better Never than Late : Meeting Deadlines in Datacenter Networks," *ACM SIGCOMM*, 2011, pp. 50–61.
[6] M. Alizadeh *et al.*, "Data Center TCP (DCTCP)," *ACM SIGCOMM*, 2010, pp. 63–74.
[7] Y. Zhang and N. Ansari, "On Mitigating TCP Incast in Data Center Networks," *IEEE INFOCOM*, 2011, pp. 51–55.
[8] "Amazon Found Every 100ms of Latency Cost Them 1% in Sales," http://blog.gigaspaces.com/amazon-found-every-100ms-of-latency-cost-them-1-in-sales/, Aug., 2008.
[9] http://glinden.blogspot.com/2006/11/marissa-mayer-at-web-20.html.
[10] "The Value of a Millisecond: Finding The Optimal Speed of a Trading Infrastructure," http://www.tabbgroup.com/PublicationDetail.aspx?PublicationID=346, 2008.
[11] G. DeCandia *et al.*, "Dynamo: Amazon's Highly Available Key-value Store," *ACM SIGOPS Operating Systems Review*, vol. 41, no. 6, 2007, pp. 205–20.
[12] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-Aware Datacenter TCP (D2TCP)," *ACM SIGCOMM*, 2012.
[13] C. Hong, M. Caesar, and P. Godfrey, "Finishing Flows Quickly with Preemptive Scheduling," *ACM SIGCOMM*, 2012.
[14] D. Zats *et al.*, "DeTail: Reducing the Flow Completion Time Tail in Datacenter Networks," *ACM SIGCOMM*, 2012.
[15] M. Alizadeh *et al.*, "Deconstructing Datacenter Packet Transport," *ACM Wksp. HotNet*, 2012, pp. 133–38.
[16] J. Ousterhout *et al.*, "The Case for RAMClouds: Scalable High-Performance Storage Entirely in DRAM," *ACM SIGOPS Operating Systems Review*, vol. 43, no. 4, 2010, pp. 92–105.
[17] G. Wang and T. Ng, "The Impact of Virtualization on Network Performance of Amazon EC2 Data Center," *IEEE INFOCOM*, 2010, pp. 1–9.
[18] A. Kangarlou *et al.*, "vSnoop: Improving TCP Throughput in Virtualized Environments via Acknowledgement Offload," *ACM/IEEE Supercomputing*, 2010, pp. 1–11.
[19] S. Gamage *et al.*, "Opportunistic Flooding to Improve TCP Transmit Performance in Virtualized Clouds," *ACM Symp. Cloud Computing (SOCC)*, 2011.
[20] V. Jeyakumar *et al.*, "EyeQ: Practical Network Performance Isolation for the Multi-tenant Cloud," *USENIX HotCloud*, 2012, pp. 1–8.
[21] A. Shieh *et al.*, "Sharing the Data Center Network," USENIX NSDI, 2011.
[22] L. Popa *et al.*, "Faircloud: Sharing the Network in Cloud Computing," *ACM SIGCOMM*, 2012, pp. 187–98.
[23] J. C. Mogul and L. Popa, "What We Talk About When We Talk About Cloud Network Performance," *ACM SIGCOMM CCR*, vol. 42, no. 5, 2012, pp. 44–48.
[24] C. Raiciu *et al.*, "Improving Datacenter Performance and Robustness with Multipath TCP," *ACM SIGCOMM*, 2011, pp. 265–76.
[25] D. Crisan *et al.*, "Short and Fat: TCP Performance in CEE Datacenter Networks," *IEEE Symp. High Performance Interconnects*, 2011, pp. 43–50.

Biographies

JIAO ZHANG (zhangjiao1986@gmail.com) is currently a Ph.D. candidate of the Department of Computer Science and Technology, Tsinghua University, Beijing, China. Her supervisor is Prof. Fengyuan Ren. She received her Bachelor's degree in computer science and technology from Beijing University of Posts and Telecommunication in 2008. Since August 2012, she has been a visiting student in the networking group of ICSI, University of California, Berkeley. Her recent research focuses on traffic management in data center networks. She has also done some work on data aggregation and energy-efficient routing in wireless sensor networks before.

FENGYUAN REN [M] (renfy@tsinghua.edu.cn) is a professor of the Department of Computer Science and Technology at Tsinghua University. He received his B.A and M.Sc. degrees in automatic control from Northwestern Polytechnic University, China, in 1993 and 1996, respectively. In December 1999, he obtained his Ph.D. degree in computer science from Northwestern Polytechnic University. From 2000 to 2001, he worked in the Electronic Engineering Department of Tsinghua University as a postdoctoral researcher. In January 2002, he moved to the Computer Science and Technology Department of Tsinghua University. His research interests include network traffic management and control, control in/over computer networks, wireless networks, and wireless sensor networks. He has (co)-authored more than 80 international journal and conference papers. He has served as a Technical Program Committee member and local arrangement chair for various IEEE and ACM international conferences.

CHUANG LIN [SM] (chlin@tsinghua.edu.cn) is a professor of the Department of Computer Science and Technology at Tsinghua University. He is an Honorary Visiting Professor, University of Bradford, United Kingdom. He received his Ph.D. degree in computer science from Tsinghua University in 1994. His current research interests include computer networks, performance evaluation, network security analysis, and Petri net theory and its applications. He has published more than 300 papers in research journals and IEEE conference proceedings in these areas, and has published four books. He is the Chinese Delegate in TC6 of IFIP. He served as Technical Program Vice Chair of the 10th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS 2004); and General Chair, ACM SIGCOMM Asia Workshop 2005 and the 2010 IEEE International Workshop on Quality of Service. He is an Associate Editor of *IEEE Transactions on Vehicular Technology*, and an Area Editor of *Computer Networks* and the *Journal of Parallel and Distributed Computing*.