Datacenter Traffic Control: Understanding Techniques and Tradeoffs

Mohammad Noormohammadpour, *Student Member, IEEE*, and Cauligi S. Raghavendra, *Fellow, IEEE*

***Abstract*—Datacenters provide cost-effective and flexible access to scalable compute and storage resources necessary for today's cloud computing needs. A typical datacenter is made up of thousands of servers connected with a large network and usually managed by one operator. To provide quality access to the variety of applications and services hosted on datacenters and maximize performance, it deems necessary to use datacenter networks effectively and efficiently. Datacenter traffic is often a mix of several classes with different priorities and requirements. This includes user-generated interactive traffic, traffic with deadlines, and long-running traffic. To this end, custom transport protocols and traffic management techniques have been developed to improve datacenter network performance. In this tutorial paper, we review the general architecture of datacenter networks, various topologies proposed for them, their traffic properties, general traffic control challenges in datacenters and general traffic control objectives. The purpose of this paper is to bring out the important characteristics of traffic control in datacenters and not to survey all existing solutions (as it is virtually impossible due to massive body of existing research). We hope to provide readers with a wide range of options and factors while considering a variety of traffic control mechanisms. We discuss various characteristics of datacenter traffic control, including management schemes, transmission control, traffic shaping, prioritization, load balancing, multipathing, and traffic scheduling. Next, we point to several open challenges as well as new and interesting networking paradigms. At the end of this paper, we briefly review inter-datacenter networks that connect geographically dispersed datacenters, which have been receiving increasing attention recently and pose interesting and novel research problems. To measure the performance of datacenter networks, different performance metrics have been used, such as flow completion times, deadline miss rate, throughput, and fairness. Depending on the application and user requirements, some metrics may need more attention. While investigating different traffic control techniques, we point out the tradeoffs involved in terms of costs, complexity, and performance. We find that a combination of different traffic control techniques may be necessary at particular entities and layers in the network to improve the variety of performance metrics. We also find that despite significant research efforts, there are still open problems that demand further attention from the research community.**

***Index Terms*—Datacenters, traffic control, tradeoffs, management schemes, transmission control, traffic shaping, load balancing, flow prioritization, multipath routing, traffic scheduling.**

Manuscript received May 12, 2017; revised August 13, 2017, October 7, 2017, and November 23, 2017; accepted December 10, 2017. Date of publication December 14, 2017; date of current version May 22, 2018. *(Corresponding author: Mohammad Noormohammadpour.)*

The authors are with the Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089 USA (e-mail: noormoha@alumni.usc.edu; raghu@usc.edu).

Digital Object Identifier 10.1109/COMST.2017.2782753

I. Introduction

DATACENTERS provide an infrastructure for many online services such as on-demand video delivery, storage and file sharing, Web search, social networks, cloud computing, financial services, recommendation systems, and interactive online tools. Such services may dynamically scale across a datacenter according to demands enabling cost-savings for service providers. Moreover, considering some degree of statistical multiplexing, better resource utilization can be achieved by allowing many services and applications to share datacenter infrastructure. To reduce costs of building and maintaining datacenters, numerous customers rely on infrastructure provided by large cloud companies [1]–[3] with datacenters consisting of hundreds of thousands of servers.

Figure 1 shows the structure of a typical datacenter cluster network with many racks. A datacenter often hosts multiple such clusters with thousands of machines per cluster. A cluster is usually made up of up to hundreds of racks [4]–[6]. A rack is essentially a group of machines which can communicate at line rate with minimum number of hops. All the machines in a rack are connected to a **Top of Rack (ToR)** switch which provides non-blocking connectivity among them. Rack size is typically limited by maximum number of ports that ToR switches provide and the ratio of downlink to uplink bandwidth. There is usually about tens of machines per rack [4]–[6]. ToR switches are then connected via a large interconnection allowing machines to communicate across racks. An ideal network should act as a huge non-blocking switch to which all servers are directly connected allowing them to simultaneously communicate with maximum rate.

Datacenter network topology plays a significant role in determining the level of failure resiliency, ease of incremental expansion, communication bandwidth and latency. The aim is to build a robust network that provides low latency, typically up to hundreds of microseconds [7]–[9], and high bandwidth across servers. Many network designs have been proposed for datacenters [5], [10]–[16]. These networks often come with a large degree of path redundancy which allows for increased fault tolerance. Also, to reduce deployment costs, some topologies scale into large networks by connecting many inexpensive switches to achieve the desired aggregate capacity and number of machines [4], [17]. Although the majority of these topologies are symmetrical, in practice, datacenter networks turn out to be often asymmetrical due to frequent failures of network elements (switches, links, ports, etc.) [18]–[20]. In contrast to fixed networks, reconfigurable topologies involve optical

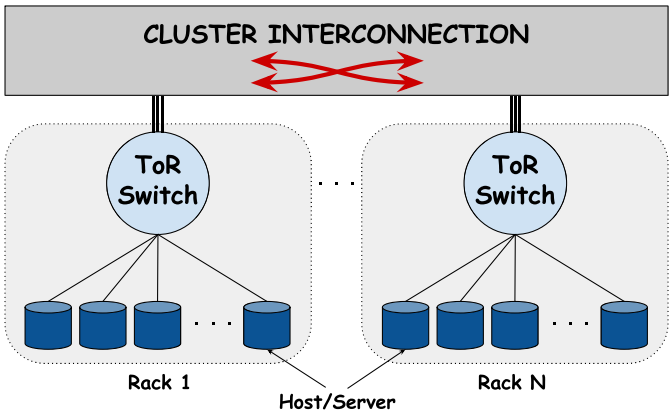


Fig. 1. A typical datacenter cluster.

circuit switching, wireless or a combination of both to adapt to traffic demands [21]–[24]. These topologies rely on fast algorithms that take into account the reconfiguration latency.

Many applications may need to span over multiple racks to access required volume of resources (storage, compute, etc.). This increases the overall volume of traffic across racks. A datacenter network with full bisection bandwidth allows for flexible operation and placement of applications across clusters and improves overall resource utilization and on-demand scale out for applications [4], [5], [10], [15]. This allows resources of any machine to be used by any application which is essential for hyper-scale cloud providers [1]–[3].

Designing networks for full bisection bandwidth is costly and unnecessary for smaller companies or enterprises. As a result, many datacenters may be over-subscribed, meaning the total inter-rack network capacity may be less than sum of intra-rack capacities across all racks. The underlying assumption is that applications are mostly rack local. Increasing the over-subscription ratio affects performance of different topologies differently. For instance, over-subscribed Fat-Trees provide less flexible communication across machines compared to over-subscribed Jellyfish networks [25].

There is growing demand for datacenter network bandwidth. This is driven by faster storage devices, rising volume of user and application data, reduced cost of cloud services and ease of access to cloud services. Google reports 100% increase in their datacenter networking demands every 12 to 15 months [4]. Cisco forecasts a 400% increase in global datacenter IP traffic and 2.6 times growth in global datacenter workloads from 2015 to 2020 [26].

Applications running on datacenters determine their traffic characteristics and the communication patterns across machines. Some popular applications include Web services, cache followers, file stores, key-value stores, data mining, search indexing and Web search. Many datacenters, especially cloud providers, run a variety of applications that results in a spectrum of workloads. Some applications generate lots of internal datacenter traffic, such as scatter-gather (also known as partition-aggregate) [27]–[30] and batch computing tasks [31], [32]. As a result, the total traffic volume within a datacenter is often much more than that of entering or leaving it. Cisco reports this ratio to be greater than 3 which is expected to increase further by 2020 [26]. Traffic control is necessary to highly utilize network bandwidth, keep latency low, offer quality of service, and fairly share resources among many users and applications by managing flow of traffic across the datacenter. There is a significant body of work on traffic control for datacenters. In this tutorial paper, we aim to review concepts in design and operation of traffic control schemes.

The rest of this paper is organized as follows. In Section II, we present some related work. In Section III, we review a variety of datacenter topologies, provide an overview of datacenter traffic patterns, set forth the challenges of traffic control for datacenter networks, and the objectives of datacenter traffic control. In Section IV, we review management schemes that can be applied to datacenters and point to some research work that use different management schemes. Next, in Section V, we present a variety of traffic control techniques, discuss them in detail and explore the benefits and drawbacks associated with them. In Section VI, we discuss some general open traffic control problems. In Section VII, we point to rising paradigms related to datacenters. In Section VIII, we introduce a new research area that is a result of global distribution of datacenters. Finally, in Section IX, we conclude the paper.

II. Related Work

In this section, we briefly present some survey articles related to datacenter traffic control. Liu *et al.* [33] provide a short survey of low latency datacenter networking by reviewing approaches taken to achieve low latency, namely reducing queuing delay, accelerating retransmissions, prioritizing mice flows and utilizing multi-path. Ren *et al.* [34] survey the methods used to address the transport control protocol (TCP) incast problem (please see Section III-C6). Chen *et al.* [35] survey bandwidth sharing in multi-tenant datacenters using techniques of static reservation, minimum guarantees and no guarantees (resources obtained in a best effort fashion). Zhang *et al.* [36] point out datacenter transport problems namely TCP incast, latency, challenges in virtualized environments, bandwidth sharing in multi-tenant environments, under-utilization of bandwidth, and TCP in lossless Ethernet environments also known as Converged Enhanced Ethernet (CEE). Yadav [37] discuss TCP issues in datacenters pointing to TCP incast, queue buildup and buffer pressure. Wang *et al.* [38] provide a comprehensive overview of datacenter networking for cloud computing discussing cloud computing network architectures, communication technologies used, topologies, routing and virtualization approaches. Joglekar and Game [39] discuss various congestion notifications for datacenter networks. Finally, Sreekumari and Jung [40] survey transport protocols proposed for datacenter networks and briefly explain each one of the variety of research efforts made in addressing the incast problem, outcast problem and in reducing latency for datacenter networks.

In this tutorial paper, we merely focus on traffic control concepts that can be applied to a variety of transport protocols

including TCP. We also point to research efforts that use different techniques as examples so that readers can elevate their knowledge in case they are interested in further details. We try to uncover the (not obvious) tradeoffs in terms of complexity, performance and costs. This paper is different from prior work in that it covers a variety of aspects in traffic control in a conceptual way while not focusing on any specific transport, network or data link layer protocol. In addition, we provide an overview of datacenter traffic properties, topologies as well as traffic control challenges, objectives and techniques and their relationships which has not been done in prior work to our knowledge. Finally, at the end of this paper, we point to a recent research direction that involves inter-datacenter networks and offer three areas that demand further attention from the research community.

III. Datacenter Networks

In this section, we dive deeper into specifics of datacenter networks. We review a variety of topologies proposed and general traffic properties of datacenter networks, point to traffic control challenges in datacenters and explain several traffic control objectives sought by various parties (i.e., operators, tenants, etc.) involved in using datacenter networks.

A. Topologies

We shortly review popular physical datacenter topologies proposed and used in the literature either in testbed or simulation. Figure 2 shows examples of datacenter network topologies reviewed in the following (notice the network connecting ToR switches).

1) Fat-Tree: Fat-Tree [10], shown in Figure 2(a), is a multi-rooted tree topology where every root is called a core switch. It can be considered as a folded Clos [17] network [42]. By increasing the number of roots, one can reduce the over subscription ratio (considering fixed capacity per link). This topology allows for high bisection bandwidth using a large number of less expensive switches allowing support for a large number of hosts at much less cost. There is an aggregate layer between the core and edge (ToR switches). The number of hops between any two servers attached to the same ToR switch is 2, to the same aggregate switch is 4 and otherwise is 6. A Fat-Tree topology built with k-port switches supports up to $\frac{k^3}{4}$ physical servers (assuming one physical port per server) and $\frac{k^2}{4}$ paths between any source and destination pair. As a result, it is possible to scale to huge clusters by interconnecting many inexpensive switches.

To effectively use a Fat-Tree, complex routing configurations may need to be done on switches to avoid creation of loops while using available paths for load balancing. For example, Portland [43] is a custom routing and forwarding protocol which works out of Layer 2 and improves on fault tolerance (link failures), scalability, and ease of management (e.g., moving VMs across physical servers). Portland uses a Fabric Manager that holds information on address mappings and a fault matrix that maintains per link health status.

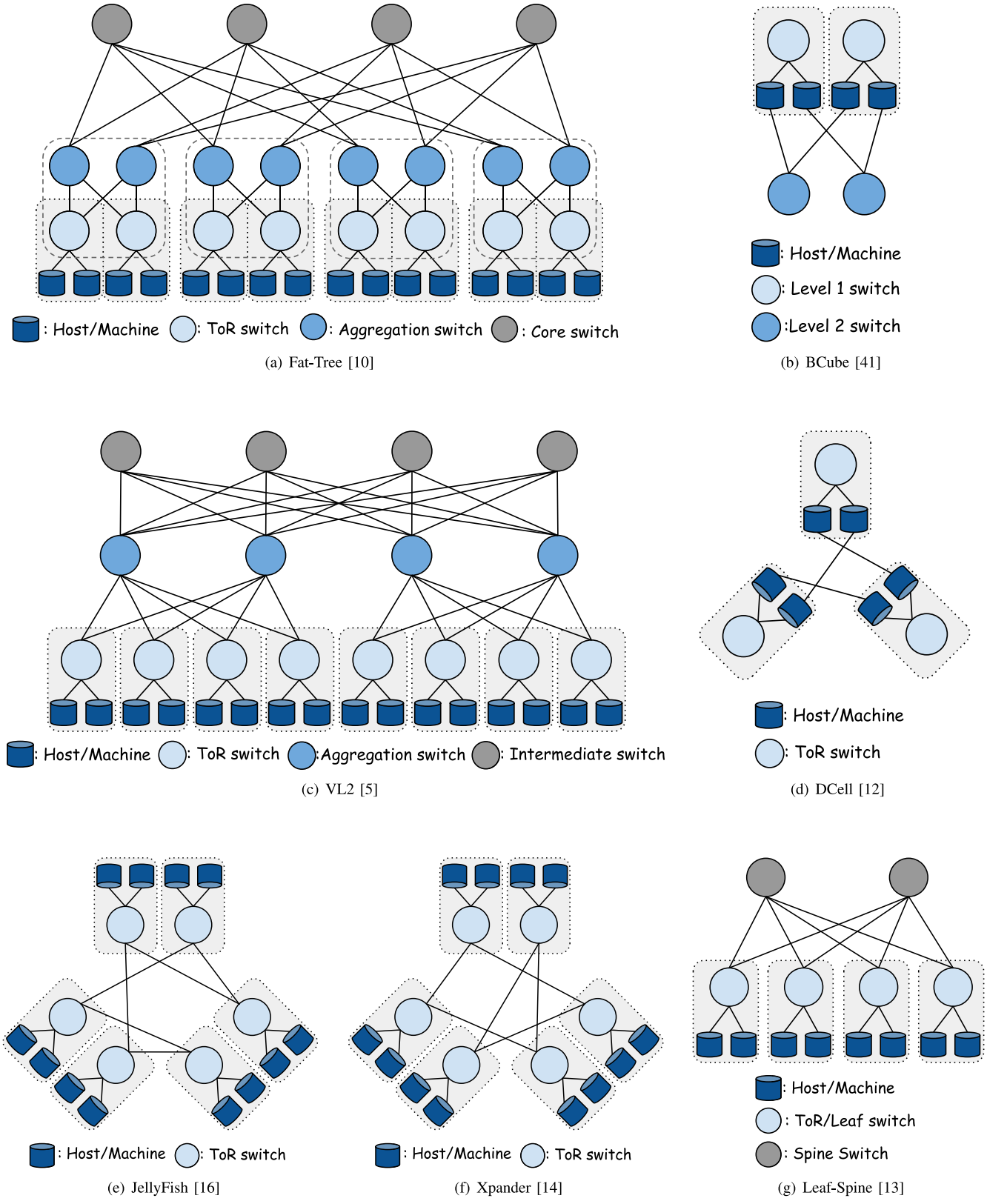
2) Leaf-Spine: Leaf-Spine (or Spine-and-Leaf) [13], [44], shown in Figure 2(g), is a two tier network topology where leaf (i.e., ToR) switches are attached to servers and every spine switch is directly connected to all leaf switches similar to a bipartite graph. The links connected between the servers and leaf switches may have a different capacity from the ones connecting leaf switches to the spine. Leaf-Spine makes it easier to expand on capacity and ports (by adding more spine or leaf switches) and also allows straight-forward usage of Layer 3 routing with load balancing support without creation of loops. As a downside, in case high bisection bandwidth is intended, scaling to more than hundreds of servers in a cluster can lead to increased costs due to need for spine switches with many high capacity ports.

3) VL2: VL2 [5], shown in Figure 2(c), implements a complete routing and forwarding suite on top of 3-tier folded Clos networks (a multi-rooted tree) but differs from Fat-Tree in that switch-to-switch links are assigned much higher capacity than server-to-switch links. This requires less number of cables for connecting the aggregation and core layer. This topology has an intermediate tier that is connected to the aggregation trier in a bipartite graph topology. Each edge (ToR) switch is connected to two aggregation switches in a way that each aggregation switch gets connected to equal number of edge switches.

4) JellyFish: JellyFish [16], shown in Figure 2(e), is a topology where ToR switches are connected in a random setting and according to some rules: first, ports are randomly connected between switches until no more links can be added; next, for any switch S with two or more free ports, an existing link $A-B$ is removed and two links are added between two of S's free ports and two ends of the removed network link (i.e., $A-S$ and $B-S$), until no switch has more than one free port [16]. Since ToR switches are connected directly, the average path length (number of hops) is considerably smaller compared to 3-tier folded Clos. In addition, this topology is much easier to expand gradually. Authors show that with full bisection bandwidth, JellyFish supports more servers at the same cost compared to Fat-Tree. Also, with the same number of failed links, JellyFish offers a higher average throughput per server than Fat-Tree. One major problem with this topology is that the existing routing schemes cannot effectively use all paths since the number of hops across parallel paths changes by a large degree. Authors propose usage of k-shortest path routing for flows along with multipath TCP (MPTCP) [45].

5) DCell: DCell [12], shown in Figure 2(d), is a hierarchical datacenter topology where a higher level DCell is built by putting together multiple lower level DCell structures. It can be incrementally expanded and does not have a single point of failure. DCell uses a custom routing algorithm that takes into account failures (DCell Fault-tolerant Routing) while aiming for near shortest path routing. Authors show that DCell offers higher network capacity compared to conventional tree topologies; however, it can perform much worse than multi-rooted trees such as Fat-Tree [41]. Implementing DCell requires changes to the server networking protocol stack.

6) BCube: BCube [41], shown in Figure 2(b), is a leveled structure where a higher level is built by recursively attaching lower levels. Servers require multiple network ports to connect



(a) Fat-Tree [10]

(b) BCube [41]

(c) VL2 [5]

(d) DCell [12]

(e) JellyFish [16]

(f) Xpander [14]

(g) Leaf-Spine [13]

Fig. 2. Several examples of datacenter topologies with two machines per rack.

to switches and they also act as forwarding elements for other servers. BCube uses source routing and requires changes to the server networking protocol stack which can be done either in hardware (network card) or software. If forwarding is implemented in software, it can add CPU overhead especially at high rates. Authors show that BCube is more resilient to switch

failures compared to Fat-Tree and almost as resilient to server failures.

7) Xpander: Xpander [14], shown in Figure 2(f), is a datacenter topology based on expander graphs [46] which offers all the performance improvements of JellyFish over Fat-Tree topology with higher throughput as network is incrementally expanded. This topology is made by connecting multiple meta nodes based on the following rules: first, each meta node is made up of equal number of ToR switches; second, no two ToRs are connected within the same meta node; third, same number of links is used to connect every pair of meta nodes. Compared to JellyFish, Xpander offers the benefit of being structured rather than random which improves implementation predictability. Authors test Xpander topology with similar routing and forwarding techniques used in JellyFish: *k*-shortest paths routing and MPTCP for multi-path load distribution.

B. Traffic Properties

Traffic characteristics of datacenters is highly dependant on applications and determines distribution of flow arrivals, flow sizes, and flow durations. For example, flows generated by Web search queries are usually much smaller and shorter than that of batch computing jobs. This variety of applications could cause creation of long lived connections as well as short microbursts on the same network [47]. There are limited publications on datacenter traffic characteristics. We review these works briefly focusing on applications and flow properties.

Kandula *et al.* [27] collect a traffic dataset from a cluster running query processing applications on large volumes of data that run MapReduce like workloads under the hood to respond to queries. Authors find that more than 50% of flows last less than 100 ms, 80% of flows last less than 10 seconds, and almost all flows last less than 100 seconds. They also find that more than 50% of traffic is carried by flows that last less than 25 seconds, less than 10% by flows that last less than 10 seconds, and almost 1% of traffic by flows that last less than 1 second. In terms of flow arrivals, authors find periodic short-term bursts of flows and periods of long silence.

In another work [48], Benson *et al.* collect and process datasets from 19 datacenters with a wide range of workloads including Web services, instant messaging and MapReduce. This work is focused on packet level traffic analysis and does not offer statistics on a per-flow basis. Authors observed an ON/OFF pattern for packet arrivals at ToR switches they monitored, meaning there was varying periods with no packet arrivals between bursts of packets.

Benson *et al.* [49] study 10 datacenters categorized as educational (storage, email, Web, video), enterprise (custom applications in addition to storage, Web and email) and cloud (instant messaging, Web, search, indexing, video). They report the number of active flows less than 10000 per second per rack across all datacenters. More than 40% of flows were less than 1 KB, more than 80% of were less than 10 KB, and almost all flows were less than 10 MB. According to their results, durations are more dependant on the specific datacenter: the smallest median flow duration was about 500 μs in one datacenter while the largest median flow duration was 1 second in a different datacenter. The largest flow duration was between 50 seconds and 200 seconds across all datacenters. This work also confirms the ON/OFF behavior of packet arrivals.

In a recent paper [50], Facebook shares statistics on their traffic characteristics. They report flow size distributions on a per application basis for three major applications they run. Median and tail flow sizes for Hadoop, Web Server and Cache applications are reported to be between about 100 KB and 100 MB, 3 KB and 10 MB, 1 KB and 10 KB within racks while 1 KB and 1 MB, 5 KB and 500 KB, 30 KB and 3 MB between racks, respectively. Regarding flow durations, Hadoop flows had a median of about 300 ms and tail of less than 1000 seconds, Web Server flows had a median of about 900 ms and a tail of about 200 seconds, and Cache flows had a median of almost 400 seconds and a tail of almost 800 seconds, respectively. Per server, the median inter arrival time of various flow types was between 1000 μs and 10000 μs and the tail was between 10000 μs and 100000 μs. Finally, authors did not observe an ON/OFF packet arrival pattern at the switches which is suggested to be due to a large number of concurrent destinations, since ON/OFF pattern was observed on a per destination basis.

In addition to per-flow properties, since racks constitute a main unit in datacenter networks, one may be interested in how much traffic stays within racks. This could be helpful in deciding the over subscription ratio of cluster interconnection. The ratio of rack local to inter-rack traffic is dependent on applications that run within the datacenter and how instances of such applications are deployed on the servers. As a result, some prior work report a highly rack local traffic pattern [49] while some find traffic neither rack local nor all to all [50], i.e., for some applications (e.g., Hadoop) traffic is mainly rack local while for others (e.g., Web Server and Cache) traffic is not at all rack local.

In summary, traffic characteristics, such as packet sizes, flow size distributions and flow inter-arrival times are highly correlated with applications. In addition, locality of traffic to racks is highly dependant on the way applications are deployed across the network and how such applications communicate with one another. For example, Roy *et al.* [50] report that servers of different types are deployed in separate racks and since most communication occurs between Web Servers and Cache Followers (different server types), there is less rack locality for these clusters. Given such strong dependency on applications, it is relatively hard to draw a general conclusion about datacenter traffic. Some common findings include the following. There is usually several orders of magnitude difference between median and maximum flow sizes (the actual difference varies according to applications). In addition, there can be a large number of flow arrivals per server (as many as thousands per second) with many concurrent flows. Finally, distributions of flow sizes and durations may be considerably different due to possibly uneven distribution of traffic across the network (flows may have to compete for bandwidth) and application of techniques like connection pooling which leads to long-lived connections that are not always transmitting.

TABLE I
SUMMARY OF DATACENTER TRAFFIC CONTROL CHALLENGES

Challenge	Description	Implications
Unpredictable Traffic Matrix §III-C1	A traffic matrix represents the communication volume between pairs of end-points in a computer network. In datacenters, traffic matrix is varying and unpredictable.	Complicates traffic engineering and capacity planning.
Mix of Flow Types and Sizes §III-C2	Due to variety of applications that share the datacenter infrastructure, a mix of various flow types and sizes are generated. Flows may have deadline constraints or not and may be short latency-sensitive or large throughput-oriented. Also, for specific applications, flow sizes may be unknown.	Complicates flow scheduling to meet requirements of different flows over a shared medium.
Traffic Burstiness §III-C3	Burstiness has two different aspects. Traffic per flow could be highly bursty and flow arrival itself could be bursty as well. Burstiness intensity may change according to where traffic is measured, i.e., at end-point interfaces, at ToR switch ports, and so on.	Large buffer space at the switches to absorb bursts, careful and responsive traffic control to minimize average buffer space usage and react to bursts quickly.
Packet Reordering §III-C4	Can be caused while applying some traffic control schemes to improve load balancing or increase throughput via using parallel paths at the same time. At high rates, reordering can exhaust end-point CPU and memory resources for buffering and putting segments back in order.	Efficient methods to put packets in order at receivers with minimal memory and CPU overhead and careful transmission methods at senders to minimize reordering when packets arrive at the receiver.
Performance Isolation §III-C5	In cloud environments with many tenants where network resources are shared across tenants, mechanisms should be put in place to make sure tenants' use of resources cannot impact other tenants.	Allocation of bandwidth on a per tenant basis rather than per flow taking into account possibility of selfish or malicious behavior from tenants.
The Incast Problem §III-C6	A variety of applications, such as search, use the partition-aggregate communication pattern which can lead to a large number of incoming flows to end-points. If not properly handled, this can lead to congestion, dropped packets and increased latency.	Larger available switch buffer space, responsive and careful traffic control to keep switch buffer occupancy low and avoid dropped packets.
The Outcast Problem §III-C7	Is observed in switches that use DropTail queues and when a disproportionate number of flows arrive at different incoming ports and exit the same switch output port. This problem is caused by synchronous dropped packets under high utilization.	Responsive and careful traffic control to keep switch buffer occupancy low and avoid dropped packets.

C. Traffic Control Challenges

We present some datacenter traffic control challenges frequently pointed to in the literature. Table I provides a summary of these challenges.

1) Unpredictable Traffic Matrix: A variety of applications usually run on a datacenter creating many flows with different properties. Most datacenter flows are short[1] (a few packets at most) [5], [7], [27], [29], [48]–[50], [53] and many short flows may be only one packet long [53]. However, most bytes are delivered by large flows [5], [7], [29]. Reference [5] reports 80% of flows to be less than 10 KB and 99% of flows to be less than 100 MB.

Datacenter applications also determine the flow arrival rates and distributions. The median flow inter-arrival time for a single machine was reported between 2 ms to 10 ms (100 to 500 flows per second) for different servers in Facebook datacenters [50] (this median is calculated by measuring number of arriving flows per machine per second as samples averaged over number of machines). Reference [49] reports between 100 to 10000 flow arrivals per switch in a one second bin in different educational and private datacenters. Finally, [27] finds the median arrival rate of flows in a cluster to be 100 flows per millisecond.

High flow arrival rate majority of which being short can create an unpredictable and fluctuating traffic matrix which makes it hard to perform traffic engineering or capacity planning in longer time scales to improve performance [5], [27], [54], [55].

2) Mix of Various Flow Types/Sizes: Traffic in datacenters is a mix of various flow types and sizes [5], [29], [48], [49], [51], [56]. Knowledge of various flow requirements and characteristics can help us design and tune transport protocols to more efficiently use network resources. Size and priority of a flow are usually determined by the application that initiates it. For some applications, flow sizes maybe unknown upon initiation.

Interactive flows which are created as a result of user interactions (for example generated by soft real time applications such as Web search) can generate latency-sensitive flows that are usually short and should be delivered with high priority. Examples include queries (2 to 20 KB) and short messages (100 KB to 1 MB) [29]. Size of these flows is usually known apriori [51]. Responsiveness of online services depends on how interactive flows are handled which can impact the number of users for an online service in the long run. In a study by Google, 400 ms increase in delay reduced the number of searches by 0.6% per user per day [57]. Also, 100 ms added latency could reduce Amazon sales by 1% [58].

Throughput-oriented flows are not as sensitive to delay, but need consistent bandwidth [29]. They may range from moderate transfers (1 MB to 100 MB) such as ones created by data parallel computation jobs (e.g., MapReduce), to background long-running flows that deliver large volumes of data such as transfers that move data across datacenter sites for

[1]Short flows are usually considered to be less than 1 MB [29], [51], [52].

data warehousing and geo-replication [59]. For these flows, it is still preferred to minimize the transfer time.

Deadline flows have to be completed prior to some deadlines. Their size is either known [56] or a good estimate can typically be drawn [51]. Both latency sensitive and throughput oriented flows might have deadlines. Deadlines can be either soft or hard which implies how value of its completion drops as time passes [60]. A soft deadline implies that it is still profitable to complete the flow and that the value decreases according to a utility function as we move past and away from the deadline. A hard deadline means zero value once the deadline has passed. For example, in interactive scatter-gather applications, if a query is not replied by its deadline (usually less than 300 ms [29], [61]), the final answer has to be computed without it [62] (i.e., zero value for that flow), while if a backup process is not completed in time, it is still valuable to finish it, although it might increase the risk of user data loss due to failures.

3) Traffic Burstiness: Several studies find datacenter traffic bursty [9], [29], [48]–[50], [63]. Theoretically, bursty traffic has been shown to increase packet loss and queuing delay while deceasing throughput [64]. In bursty environments, packet losses have been found more frequent at the network edge due to higher burstiness [48]. Burstiness can also lead to higher average queue occupancy in the network leading to increased flow completion times (FCT) for many flows [9] and increased packet drops due to temporary creation of full buffers in the network [29], [65]. In addition, highly bursty traffic can cause buffer space unfairness in shared memory switches if a switch port exhausts shared memory due to receiving long bursts [29].

Several causes can lead to creation of bursty traffic. Hardware offloading features, such as Large Send Offload (LSO), that reduce CPU load, can lead to higher burstiness. Interrupt Moderation (Coalescing) [66], which reduces CPU load and increases throughput by processing packets in batches, can lead to bursty traffic at the sender. Transport control features in software can create bursty traffic by scheduling a large window of packets to be sent together such as TCP slow start. Applications can increase burstiness by sending large pieces of data at once [63].

4) Packet Reordering: Out of order arrival of packets can increase memory and CPU utilization and latency at the receiver especially at high rates [67], [68]. Some transport protocol features, such as fast retransmit [69], may mistake reordering with packet loss. Different hardware and software features are involved in putting packets back in order including Large Receive Offloading (LRO) [70], Receive Side Scaling (RSS) [71] and Generic Receive Offloading (GRO) [72]. LRO and GRO are usually implemented as part of driver software. Some NICs provide hardware support for these features. LRO focuses mostly on TCP/IPv4 stack while GRO is general purpose.

To understand the extend to which reordering can affect performance, we point to a prior work on improving the performance of handling out of order packet arrivals [67]. Authors performed experiments with Vanilla Linux kernel and realized that at high rates (e.g., gigabits), significant reordering can increase CPU utilization to 100% and limit server interface link utilization to 70%. Even after applying optimizations at the driver and network stack level, CPU load increased by about 10% with server interface link utilization at 100%.

5) Performance Isolation: Performance isolation is necessary in cloud environments where multiple tenants use shared network resources [73]–[75]. Isolation prevents selfish or malicious behavior that aims to either unfairly obtain more resources, such as by creating many flows or using custom aggressive transport protocols [76], [77], or to cause disruptions.

Enforcing performance isolation over a shared infrastructure is hard. To effectively isolate the effect of tenants and users, mechanisms need to be put in place in various layers and parts of the network. For example, a queue management scheme will need to divide buffer space according to users and tenants, bandwidth needs to be fairly divided, computational and memory overheads due to network communication needs to be controlled on a per user or per tenant basis, and all of this need to be done according to service level agreements between the operator, users and tenants.

6) The Incast Problem: When many end-hosts send traffic to one destination concurrently, the link attached to the destination turns into a bottleneck resulting in queue buildups, large queuing delays, and dropped packets [29], [65], [78]–[81]. This problem becomes more serious in high bandwidth low latency networks [82] and with shallow buffer switches [83].

For example, the incast problem could appear in clusters running applications such as search and batch computing jobs like MapReduce that follow the partition-aggregate processing model. In search applications, a server may query a large number of other servers the results to which may be returned to that server at the same time creating sudden increase in incoming traffic. In a MapReduce job, a Reduce instance may download outputs of many Map instances for reduction which can cause a similar situation. Both scenarios follow the many-to-one communication pattern.

7) The Outcast Problem: This problem occurs due to synchronized drops of packets from an input port of a switch which is referred to as port blackout [84]. This eventually leads to unfairness. For such synchronous drops to happen, two predicates have to be present. First, there needs to be contention between a large group of flows coming from one input port and a small group of flows coming from a different input port for access to the same output port. Second, the output port uses queues that follow TailDrop policy (if queue is full, the arriving packet is discarded). Port blackout occurs for the small group of flows and is observed temporarily over short periods of time. When the output queue is full, any arriving packet (during this window) is dropped which leads to consecutive drops for incoming flows. Such consecutive drops affect the smaller set of flows more than they affect the larger set (a smaller number of flows in the larger set are affected). The intensity of this problem increases as the ratio of flows in the larger over smaller group increases. This problem is called the "outcast" problem because some flows are cast aside (they cannot obtain bandwidth).

TABLE II
SUMMARY OF DATACENTER TRAFFIC CONTROL OBJECTIVES

Objective	Description
Minimizing Flow Completion Times (FCT) §III-D1	Faster completion times reduces the communication delay for distributed applications and improves their end-to-end responsiveness.
Minimizing Deadline Miss Rate or Lateness §III-D2	For time constrained applications, it is important to meet specific deadline requirements. For some applications, only transactions that complete prior to their deadlines are valuable in which case deadline miss rate is the right performance metric. For some applications transactions completed past the deadlines are still valuable or even necessary in which case lateness (i.e., by how much we miss deadlines), is the right metric.
Maximizing Utilization §III-D3	To maximize performance, it is desired to use available resources as much as possible.
Fairness §III-D4	Resources should be fairly divided across tenants and users while paying attention to their class of service and service level agreements (SLAs).

This could simply occur in tree-based topologies such as Fat-Tree and in partition-aggregate communication scenarios where many flows from different servers could be returning results to one server (many-to-one communication pattern). A disproportionate number of flows from incoming servers may end up on different input ports of the ToR switch attached to the receiver which could lead to flows on some port receiving less average throughput.

D. Traffic Control Objectives

Datacenter environments involve operators, tenants, and end-users with different objectives. Operators would like to use available resources as much as possible to provide higher volume of services, accommodate more tenants and eventually increase revenues. In addition, datacenter infrastructure may be shared by several tenants running various types of applications to offer different services. Tenants would like to receive a fair share of resources according to their service level agreement (SLA) which determines the cost of services. Finally, many end-users may rely upon services offered by datacenter tenants. They would like such services to be responsive and highly available. These possibly conflicting objectives are presented in the following. Table II provides a summary of traffic control objectives in datacenters.

1) Minimizing Flow Completion Times: Flow Completion Time (FCT) is the time from the initiation of a flow to its completion. Depending on applications, FCT can directly or indirectly affect the quality of experience and service offered to end users [29], [62], [85]. Major factors that determine FCT in datacenters include queuing and packet loss [7], [29]. Occasionally, retransmission of lost segments can significantly increase latency due to the time it takes to identify and retransmit the lost data. For some applications, it may be more helpful to focus on minimizing mean or median latency (e.g., static Web applications) [29], [80], [85]–[87], while for other applications tail latency may be more important (e.g., partition-aggregate) [30], [62].

2) Minimizing Deadline Miss Rate or Lateness: Many applications require timely delivery of data that can be viewed as flows with deadlines. In such applications, as long as pieces of data are delivered prior to the deadlines, customer SLAs are satisfied. The quality of services decrease as the fraction of missed deadlines increases. For some applications, delivery after the deadlines is still valuable and necessary. As a result, sometimes minimizing the amount by which we miss deadlines is more important which is referred to as "lateness" (e.g., synchronization of search index files).

3) Maximizing Utilization: Increasing resource utilization can reduce provisioning costs and increase revenues. By better utilizing network bandwidth, operators can accommodate more tenants or improve the quality of service for existing ones. Effectively utilizing the network depends partly on network topology and design parameters, and partly on network traffic control scheme.

4) Fairness: Many flows share datacenter resources such as link bandwidth and buffer space. In multi-tenant datacenters, it is necessary to make sure tenants receive fair share of network resources according to their service level agreement (SLA). Enforcing fairness also mitigates the starvation problem and prevents malicious behavior.

There are several definitions of fairness in networking context including max-min fairness, proportional fairness, and balanced fairness [88], [89]. Fairness criteria determine how link bandwidth or buffer space is divided among flows. Max-Min Fairness (MMF) [90], which aims at maximizing the minimum share, is the most widely used. In general, fairness can be considered over multiple dimensions each representing a different kind of resource (e.g., bandwidth, CPU cycles, and memory) [91], [92]. We however focus on network bandwidth fairness in this paper.

Fairness should be considered among proper entities as defined by the fairness policy. For example, fairness may be across groups of flows as opposed to individual flows to prevent tenants from obtaining more resources by creating many flows. Strict fairness across all flows can also lead to increased number of missed deadlines [51] and sub-optimal FCTs [86]. One approach is to first apply fairness across tenants according to their classes of service and then across flows of each tenant considering flow priorities.

In addition to the traffic control objectives we mentioned, there are other objectives followed by many datacenter operators. An important objective is reducing energy costs by increasing energy efficiency. Since datacenter networks usually connect a huge number of servers, they are made of a large number of network equipment including fiber optics cables and switches. Due to varying amount of computational

TABLE III
SUMMARY OF DATACENTER TRAFFIC CONTROL MANAGEMENT

Scheme	Benefits	Drawbacks
Distributed §IV-A	Higher scalability and reliability. Solutions can be completely end-host based or use explicit network feedback. More information in §IV-A1.	Access limited to local view of network status and flow properties. Limited coordination complicates enforcement of network-wide policies. More information in §IV-A2.
Centralized §IV-B	Access to global view of network status and flow properties. Central control and management can improve flexibility and ease of enforcing network-wide policies. More information in §IV-B1.	A central controller can become a single point of failure or a network hotspot. Latency overhead of communicating with a central entity and control plane inconsistencies are other concerns. More information in §IV-B2.
Hybrid §IV-C	Potentially higher reliability, scalability, and performance. More information in §IV-C1.	Higher complexity. Also, the final solution may not be as good as a fully centralized system. More information in §IV-C2.

and storage load, average network load may be considerably less than its peak. As a result, operators may reduce energy costs by turning off a fraction of networking equipment at off-peak times [93]–[95] or by dynamically reducing the link bandwidths across certain links according to link utilizations [96]. There are however several challenges doing so. First, the resulting system should be fast enough to increase network capacity as computational or storage load increases to prevent additional communication latency. Second, it may be unclear where to reduce network capacity either by turning equipment off or by reducing link bandwidths (correlation between network load and placement of computation/storage can be considered for additional improvement [97]). Third, the effectiveness of such systems depends on load/utilization prediction accuracy. Further discussion on reducing datacenter power consumption is beyond of the scope of this paper.

IV. DATACENTER TRAFFIC CONTROL MANAGEMENT

To enforce traffic control, some level of coordination is needed across the network elements. In general, traffic control can range from fully distributed to completely centralized. Here we review the three main approaches used in the literature namely distributed, centralized or hybrid. Table III provides an overview of traffic control management schemes.

A. Distributed

Most congestion management schemes coordinate in a distributed way as it is more reliable and scalable. A distributed scheme may be implemented as part of the end-hosts, switches, or both. Some recent distributed traffic control schemes include those presented in [8], [78], [86], [98], and [99].

Designs that can be fully realized using end-hosts are usually preferred over ones that need changes in the default network functions or demand additional features at the switches such as custom priority queues [53], in-network rate negotiation and allocation [86], complex calculations in switches [100], or per flow state information [101]. End-host implementations are usually more scalable since every server handles its own traffic. Therefore, popular transport protocols rely on this type of implementation such as [29] and [102].

As an example, the incast problem Section III-C6, which is a common traffic control challenge, can be effectively addressed using end-host based approaches considering that incast congestion occurs at receiving ends. Some approaches are Server-based Flow Scheduling (SFS) [79], pHost [80], NDP [103] and ExpressPass [104]. SFS uses the generation of ACKs to control the flow of data towards receivers and avoid congestion. The sender looks at the flows it is sending and schedules the higher priority flows first, while the receiver controls the reception by deciding on when to generate ACKs. pHost uses a pull-based approach in which the sender decides on reception schedule based on some policy (preemptive or non-preemptive, fair sharing, etc). A source dispatches a Request To Send (RTS) to a receiver. The receiver then knows all the hosts that want to transmit to it and can issue tokens to allow them to send. NDP limits the aggregate transmission rate of all incast senders by maintaining a PULL queue at the receiver that is loaded with additional PULL requests when new packets arrive from a sender (a PULL request contains a counter which determines number of packets its associated sender is allowed to send). PULL requests are then sent to senders in a paced manner to make sure the overall incoming transmission rate at the receiver is not larger than per interface line rate. ExpressPass manages congestion across the network by controlling the flow of credit packets at the switches and end-hosts according to network bottlenecks (a sender is allowed to send a new data packet when it receives a credit packet).

Shifting more control to the network may allow for better resource management due to ability to control flows from more than a single server and availability of information about flows from multiple end-hosts. For example, flows from many hosts may pass through a ToR switch giving it further knowledge to perform scheduling and allocation optimizations.

Some examples of this approach include RCP [85], PDQ [86], CONGA [105], Expeditus [106], and RackCC [107]. RCP and PDQ perform in-network rate allocation and assignment by allowing switches and end-hosts to communicate using custom headers, CONGA gets help from switches to perform flowlet based load balancing in leaf-spine topologies, Expeditus performs flow based load balancing by implementing custom Layer 2 headers and localized monitoring of congestion at the switches, and RackCC uses ToR switches as means to share congestion information among many flows between the same source and destination racks to help them converge faster to proper transmission rates. To implement advanced in-network features, changes to the network elements might be necessary and switches

may need to do additional computations or support new features.

1) Benefits: Higher scalability and reliability. Can be completely implemented using end-hosts. To further improve performance, network (i.e., switches, routers, etc.) can be involved as well. Completely end-host based approaches can operate simply by controlling the transmission rate according to implicit feedbacks from network (e.g., loss, latency). Network can offer explicit feedbacks (e.g., network queues' occupancy) to improve transmission rate management by allowing senders to make more accurate control decisions. For example, Explicit Congestion Notification (ECN) allows network to communicate high queue occupancy to end-hosts [108], [109] or trimming packet payloads in case of highly occupied network queues (instead of fully discarding them) can help receivers get a complete picture of transmitted packets [103], [110].

2) Drawbacks: Usually just access to local view of network status and flow properties which allows for only locally optimal solutions. For example, while every end-point may strive to achieve maximum throughput for its flows by default (locally optimal), it may lead to a higher network wide utility if some end-points reduce their transmission rate in favor of other end-points with more critical traffic (globally optimal). Using distributed management, it may be harder to enforce new network wide policies (e.g., rate limits in a multi-tenant cloud environment) due to lack of coordination and limited view of network condition and properties.

B. Centralized

In centralized schemes a central unit coordinates transmissions in the network to avoid congestion. The central unit has access to a global view of network topology and resources, state information of switches, and end-host demands. These include flow sizes, deadlines and priorities as well as queuing status of switches and link capacities. Scheduler can proactively allocate resources temporally and spatially (several slots of time and different links) and plan transmissions in a way that optimizes the performance and minimizes contentions. To further increase performance, this entity can translate the scheduling problem into an optimization problem with resource constraints the solution to which can be approximated using fast heuristics. For large networks, scheduler effectiveness depends on its computational capacity and communication latency to end-hosts.

TDMA [111], FastPass [112] and FlowTune [113] are examples of a centrally coordinated network. TDMA divides timeline into rounds during which it collects end-host demands. Each round is divided into fixed sized slots during which hosts can communicate in a contention-less manner. All demands are processed at the end of a round and schedules are generated and given to end-hosts. FastPass achieves high utilization and low queuing by carefully scheduling traffic packet by packet considering variation in packet sizes. FlowTune improves on scalability of FastPass using centralized rate-assignment and end-host rate-control.

There are several challenges using a fully centralized approach. The central controller could be a single point of failure since all network transmissions depend on it. In case of a failure, end-hosts may have to fall back to a basic distributed scheme temporarily [112]. There will be scheduling overhead upon flow initiation, that is the time it takes for the scheduler to receive, process the request, and allocate a transmission slot. Since majority of flows are short in datacenters, the scheduling overhead has to be tiny. In addition, this approach may only scale to moderate datacenters due to processing burden of requests and creation of a congestive hot-spot around the controller due to large number of flow arrivals. Bursts in flow arrivals [27] can congest the controller temporarily and create scheduling delays. It may be possible to apply general techniques of improving scalability for central management of larger networks such as using a hierarchical design.

1) Benefits: Can provide higher performance with a global view of network status and flow properties. Such information may include utilization of different network edges, their health status as well as flows' size, deadline and priority. With this information, one can potentially direct traffic carefully according to network capacity across a variety of paths while allowing flows to transmit according to their priorities to maximize utility. Central management can improve flexibility and ease of managing network policies. For example, new routing/scheduling techniques can be rolled out much faster by only upgrading the central fabric. Centralized schemes also increase ease of admission control in case strict resource management is necessary for guaranteed SLAs.

2) Drawbacks: A central controller or management fabric can be a single point of failure or it may become a network hotspot in case of large networks. There is also latency and computational overhead of collecting network status and flow properties from many end-hosts (controller will have to process and understand incoming messages at high speed and act accordingly). Overhead of network resource allocation and scheduling (if central rate allocation is used). Finally, consistency of network updates can be an issue in large networks. For example, some updates my not be applied correctly at network elements (e.g., software bugs [20]) or different updates may be applied with varying latency that can lead to transient congestion or packet losses which may hurt performance of sensitive services.

C. Hybrid

Using a hybrid system could provide the reliability and scalability of distributed control and performance gains obtained from global network management. A general hybrid approach is to have distributed control that is assisted by centrally calculated parameters.

Examples of this approach include OTCP [114], Fibbing [115], Hedera [116] and Mahout [117]. OTCP uses a central controller to monitor and collect measurements on link latencies and their queuing extent using methods provided by Software Defined Networking (SDN) [118] which we will discuss further in Section VII-A. For every new flow, the controller calculates parameters such as initial and

minimum retransmission timeout and initial and maximum congestion window size considering which path is taken by flows which allows for fast convergence to steady state. Fibbing relies on a central controller to manipulate the costs associated with routes in the network or insert fake nodes into the routing tables of routers to force them to use or avoid some paths to control the distribution of load across network. Hedera and Mahout operate by initially allowing network to route all incoming flows using a distributed approach, then monitoring and detecting large flows that can be moved dynamically to different routes using a central controller with a global view of network status which allows for a more balanced distribution of load. While Hedera uses readily available switch capabilities (flow counters) to detect large flows, Mahout engineers and inserts a shim layer at the end-hosts to monitor and detect large flows to improve scalability.

1) Benefits: Reliability, scalability, and higher performance. A central controller can offer coarse grained solutions while fine-grained control is applied using a distributed scheme. Division of work between central and distributed components can be done in a way that minimizes the effects of centralized and distributed schemes' drawbacks. For example, in multi-tenant cloud datacenters, a hierarchical approach can be used where aggregate bandwidth given per tenant is calculated centrally while transmission control across flows per tenant is performed in a distributed fashion reducing central management overhead and increasing per tenant management scalability.

2) Drawbacks: Complexity may increase. Central controller now has limited control; therefore, the final solution may not be as good as a fully centralized system. Due to presence of centralized control, there still exists a single point of failure however with less impact in case of a failure compared to a fully centralized scheme. Also, the distributed component still operates on a locally optimal basis. For example, in multi-tenant cloud datacenters, if bandwidth per tenant is managed in a distributed fashion, due to limited local information per network element, it may be challenging to apply routing/scheduling policies that maximize utility according to flow properties.

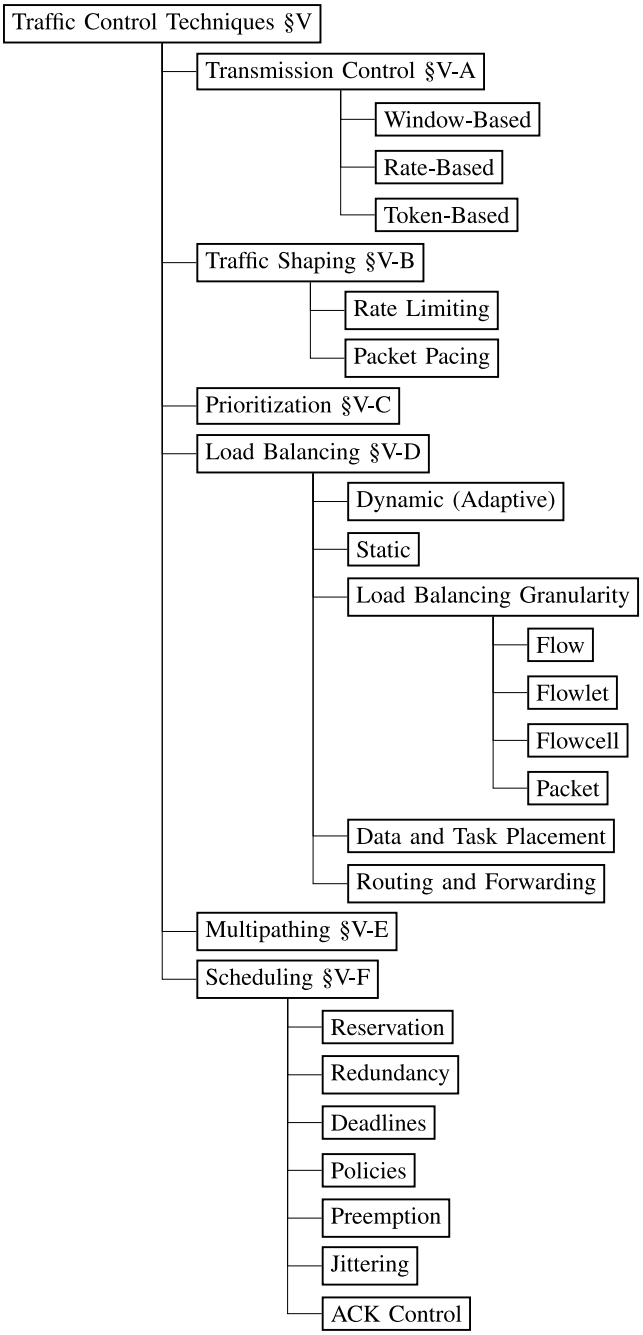


Fig. 3. High level breakdown of traffic control techniques.

V. Datacenter Traffic Control Techniques

Many design parameters and approaches can be considered in development of any traffic control scheme. Figure 3 provides a high level breakdown of major datacenter traffic control techniques. In the following, we provide a list of these techniques and discuss some research efforts made regarding each. Figure 4 provides an overview of the relationships between challenges, objectives and techniques. Details of each relationship is further explained in Tables IV and V. More detailed information is provided in the following sections.

A. Transmission Control

Transmission control is the mechanism that controls the flow of data sent to the network. There is typically a window of outstanding bytes for each flow at the sender determining the volume of data that can be transmitted before data reception is acknowledged. This allows for *implicit* rate control. A larger window may increase the average transmission rate since it allows for more bytes in flight towards receiver. A significant body of work, including recent works, employ **window-based** rate control. Table VI provides a summary of this section.

Some recent window-based approaches include DCTCP [29], D2TCP [28], L2DCT [119], MCP [120], DAQ [121], and PASE [122]. DCTCP uses explicit congestion signals from switches that is piggybacked on ACKs to change the window size according to the extent of congestion. Packets are marked to convey congestion signal according to instantaneous queue length upon their arrival at the queue.

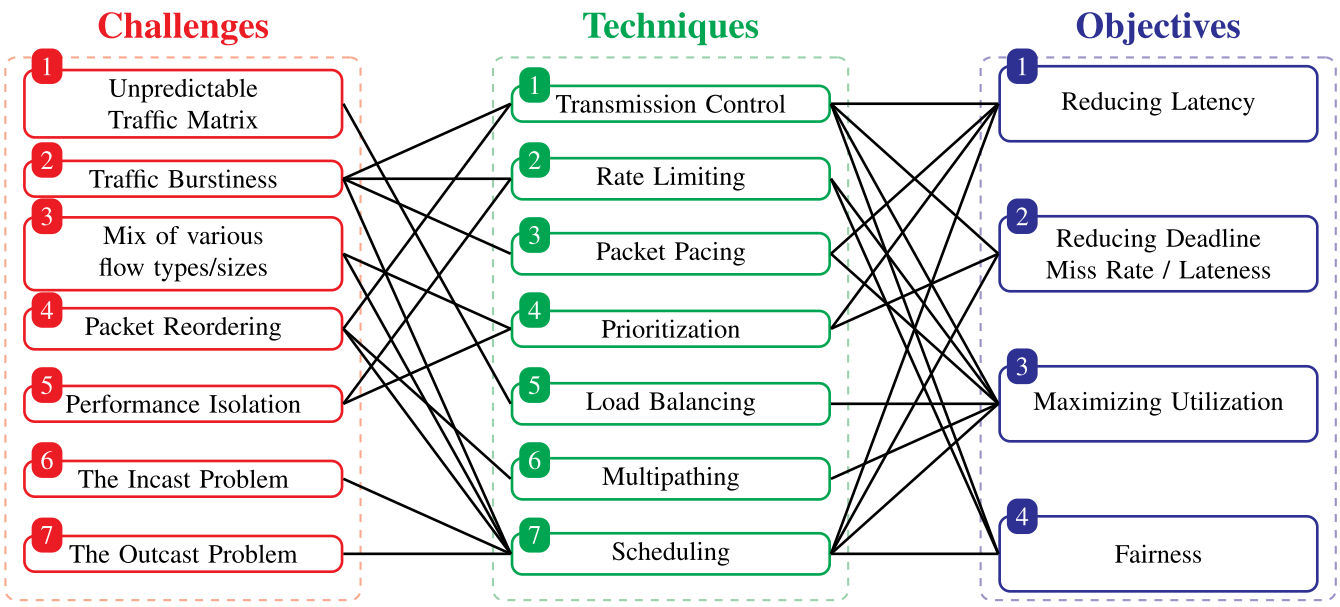


Fig. 4. How traffic control techniques interact with challenges and objectives, every line shows a direct relationship (indirect relationships may exist between some blocks but they are not shown here).

D2TCP modulates window size based on flow deadlines: flows with closer deadlines reduce window size less than others in response to network congestion signals. L2DCT modulates window size of a flow based on its estimated size which is dynamically calculated by counting the number of packets it has seen from a flow up to current time. MCP changes window size to approximate the solution to an stochastic optimization problem that minimizes long term mean packet delay using flow information and network state, such as queue length at the switches. DAQ allows ToR switches to calculate initial sender window for end-hosts to converge to proper transmission rates faster. PASE also gets help from network elements to decide on its initial coarse-grained window size, and then performs fine-grained adjustments to the window size using congestion signals received similar to DCTCP.

Although fairly simple, window-based approaches only allow for coarse-grained control of rate which can result in considerable variations and create highly bursty traffic. For example, a sender can release a full window of packets before it has to wait for the receiver to acknowledge. This may be followed by a batch of acknowledgements that move the window forward and allow for another full window of packets to be sent.

To decrease the transmission rate variations at the senders, **rate-based** mechanisms that employ *explicit* rate control can be used. Packets in the outstanding window are carefully scheduled to achieve the proper bit rate which implies reduced burstiness of the traffic. Rate-control is necessary when using reservation based bandwidth control over shared links such as in [113].

Examples of earlier rate-based protocols include TCP Friendly Rate Control (TFRC) [123] and Rate Control Protocol [85]. Several recent work also use rate-based methods in datacenters including PDQ [86], D3 [51], TIMELY [99], and RACKS [87]. TFRC calculates the allowed sending rate to compete fairly with TCP using an equation which captures various network parameters such as round trip time (RTT), loss rate, retransmission timeout, and segment size. RCP uses an equation based rate control to minimize average FCT with the help of network switches that divide bandwidth according to processor sharing policy. D3 uses flow deadlines as a factor in the rate calculation equation to reduce deadline miss rate (a closer deadline allows for a higher rate). RACS uses a similar approach to RCP but assigns weights to flows according to their priority, to emulate different scheduling algorithms at the switches according to weight assignment. PDQ improves on both RCP and D3 by rate allocation using switches and adding support for preemption. TIMELY calculates the sending rate as a function of network latency.

Explicit rate control can be applied both in hardware (e.g., NIC) and software (e.g., OS Kernel). While the former provides more precise rates, the latter is more flexible and allows for a larger number of flows to be rate controlled as well as more complex policies to be considered. Rate-control in software can result in creation of occasional bursts due to challenges of precise packet scheduling in OS kernel [123].

Complementary to these two approaches for rate control at the sender, one can also apply **token-based** transmission control which adds an extra control layer to make sure senders only transmit according to some quota assigned to them (this is sometimes referred to as *pull-based* or *credit-based* approach). For example, such quota could be assigned according to congestion status at receivers (to address the incast problem discussed earlier in Section III-C6) or to implement a form of receiver based flow scheduling policy (more information in Section V-F).

B. Traffic Shaping

We can improve network performance by making sure that it conforms to required profile and policy rules. This can

TABLE IV
DESCRIPTION OF RELATIONSHIPS BETWEEN TECHNIQUES AND CHALLENGES IN FIGURE 4

Relationship	Description	Some Related Works
1 → 2	Transmission control at senders determines number of packets given to network driver for transmission. If a bunch of packets are sent to network interface for transmission, bursts may be created.	[124], [125]
1 → 4	If transmission is performed over multiple interfaces for multipath delivery, careful transmission control is required to minimize packet reordering. This shall be done according to latencies of different paths to the receiver.	[126], [127], [128], [98]
2 → 2	Rate limiting can reduce burstiness by adding a protective layer after transmission control in case a large number of packets are given to the network driver for transmission.	[129], [130], [131]
2 → 5	Rate limiting can improve performance isolation across tenants and users by preventing any tenant/flow from taking too much bandwidth starving other tenants/flows (a tenant may initiate many flows).	[73], [132]
3 → 2	Packet pacing reduces burstiness by adding time spacing between consecutive packets prior to transmission.	[124], [125]
4 → 3	Prioritization helps allocate resources to flows accordingly and can improve the performance while scheduling a mix of flow types/sizes. For example, one could prioritize highly critical deadline flows over other flows.	[133], [134], [53], [122], [135]
4 → 5	Prioritization can shield a tenant/application with high service guarantees from other tenants/applications with lower service guarantees. Priorities shall be determined according to tenant/application service level agreements (SLAs) or Quality of Service requirements (QoS).	[136], [137]
5 → 1	Datacenter topologies usually come with a large degree of path redundancy to provide low over-subscription ratio for better inter-rack communication. With load balancing, this capacity can be used to a higher extent giving the operators/tenants a better chance to handle any traffic matrix.	[5], [10]
6 → 4	Multipathing can increase packet reordering. To reduce reordering while using multiple paths, one can perform careful packet scheduling according to path latencies at transmission control level so that packets arrive in the right order at the receiver (it is nearly impossible to eliminate reordering). Assignment of data segments to sub-flows is also an important factor in how well receivers can handle reordering.	[68], [126]
7 → 2	Transmission control, rate limiting, and packet pacing all depend on careful scheduling of packets at senders which can mitigate burstiness.	[123], [129], [130], [138]
7 → 3	When a mix of flow types is present, scheduling of packets according to flow priorities, deadlines, sizes and arrival order can help us better meet traffic control objectives.	[86], [134], [53]
7 → 4	If multiple paths are used for a flow, scheduling can reduce reordering by determining when packets should be sent at the sender over each path to arrive at the receiver with minimal reordering.	[126]
7 → 6	Scheduling can mitigate the incast problem by preventing all the incoming data (from many flows) from arriving at a receiver at the same time. Various techniques can be used such as adding random delays while initiating requests (jittering) and receiver based scheduling using either ACKs or receiver window size.	[79], [81]
7 → 7	Effective queuing disciplines which determine how packets in a switch queue are scheduled for transmission, such as Stochastic Fair Queuing (SFQ), can mitigate port blackout §III-C7.	[84]

reduce contention while using network resources. For example, traffic shaping can prevent some flows from hogging others. Shaping can be used to provide some level of resource isolation and guarantees in cloud environments with many users. Finally, traffic shaping can be useful in resource scheduling where senders follow rates specified in the schedule. A summary of traffic shaping techniques has been provided in Table VII.

1) Rate Limiting: Rate limiting is usually applied by passing traffic through a Token Bucket filter. This ensures that average transmission rate does not exceed the token generation rate. Tokens are generated at a specific rate and an arriving packet can only be sent if there is a token available. Tokens are accumulated if there is no packet to be sent, but there is usually a cap on how many. This cap is to limit traffic burstiness in case the token bucket is idle for a while. Examples of works employing rate-limiting to provide resource isolation and guarantees include [51], [73], [113], [132], [135], [140], and [141].

Rate-limiting can be done in OS Kernel, as part of the hypervisor in a virtualized environment, or via NIC. Scheduling of packets in software (Kernel or Hypervisor) is generally less precise and can be computationally intensive in high bandwidths and with a large number of flows. To reduce CPU utilization, OSes usually send packets to NIC in batches which can further reduce the scheduling precision. For example, Classful Queuing Disciplines (Qdisc) offered by Linux allows for coarse grained rate-control by determining the time and count of packets that NIC receives from RAM, however, the actual schedule of packets on the wire is determined by NIC.

To improve software rate-limiting performance one can use userspace packet processing tools some of which we point to in Section VII-D. For example, Carousel [130] is a rate-limiter that is implemented as part of a software NIC in userspace. Carousel uses a variety of techniques to reduce CPU and memory usage and improve scalability including deferred completion signalling (rate-limiter only signals completion to applications when data is actually transmitted to offer back-pressure and minimize buffer space usage) and single queue shaping (using a timing wheel and by assigning timestamps to packets over the time horizon).

Using NIC to schedule and send packets given rate limits reduces CPU load and traffic burstiness. Effective rate-limiting in hardware demands support for multiple queues and classes of traffic with hardware rate-limiters attached to them.

TABLE V
DESCRIPTION OF RELATIONSHIPS BETWEEN **TECHNIQUES** AND **OBJECTIVES** IN FIGURE 4

Relationship	Description	Some Related Works
1 → 1	Transmission control can impact overall end-to-end latency by affecting queue occupancies at the network switches.	[124], [125]
1 → 2	Transmission control determines how many packets are sent by each flow which can be done according to flow deadlines to reduce deadline miss rate and/or lateness.	[86], [28], [51]
1 → 3	Proper transmission control can maximize network bandwidth utilization while avoiding congestion (which usually leads to dropped packets and wasted bandwidth).	[112], [113], [80]
1 → 4	Transmission control plays significant role in how fairly flows share the network. For example, if one of two equally important flows is given higher transmission quota over longer periods of time, this can lead to unfairness.	[85]
2 → 3	Rate limiting can prevent congestion and reduce dropped packets. As a result, it helps maximize utilization and minimize wasted bandwidth.	[113]
2 → 4	Rate limiting can improve fairness by preventing selfish behavior of bandwidth hungry flows/tenants.	[73], [132]
3 → 1	Packet pacing can reduce average queue occupancy in the network (switches/routers) and therefore reduces end-to-end latency.	[124], [125], [9]
3 → 3	Packet pacing can increase effective utilization by preventing bursty behavior of flows which can lead to higher buffer occupancy and dropped packets. It stabilizes transmission rates and reduces transmission spikes.	[125], [9]
4 → 1	Prioritization can reduce average latency and flow completion times by giving higher priority to shorter flows.	[53], [134]
4 → 2	Prioritization according to flow deadlines can improve the overall deadline miss rate. For example, search queries with critical deadlines (e.g. 300 ms after arrival) can be given high priority and can be addressed before long running backup operations in a shared network environment.	[121]
5 → 3	Load balancing allows us to make best use of large path diversity in datacenters and maximize utilization over all available paths.	[116], [105], [98], [30]
6 → 3	Multipathing can increase utilization by allowing long running flows to use bandwidth of several available paths.	[45], [68], [139]
7 → 1	Scheduling can reduce latency by applying scheduling disciplines that mimic Shortest Remaining Processing Time (SRPT).	[87]
7 → 2	Deadline miss rate or lateness can be reduced by scheduling flows according to their deadlines such as by allotting more capacity for flows with closer deadlines.	[28]
7 → 3	Scheduling can improve utilization by reducing contention within the network for using available bandwidth. For example, by carefully deciding on when packets should be sent by end-hosts, we can avoid sudden arrival of many packets at the switches which can lead to dropped packets.	[111], [112]
7 → 4	Scheduling can improve fairness by giving bandwidth to flows based on a fair sharing policy such as Max-Min Fairness (MMF) [90].	[85], [87]

TABLE VI
OVERVIEW OF TRANSMISSION CONTROL TECHNIQUES

Scheme	Input	Benefits	Drawbacks
Window-based	Maximum outstanding window size.	Simplicity, no careful packet scheduling overhead.	Coarse-grained control over transmission rate which can lead to occasional bursts of packets (e.g. when a bunch of ACKs arrive and suddenly the transmission window moves forward).
Rate-based	Desired transmission rate (in bits per second).	More accurate control over rate, reduced variations in rate, a more natural choice for better management of network bandwidth.	Overhead of packet scheduling either in software (less accurate, less costly) or in hardware (more accurate, requires hardware support).
Token-based	Transmission quota towards a specific receiver.	Receiver enforced quota prevents senders from congesting receivers. Can be applied complementary to window/rate-based control.	A sender needs to first coordinate with the receiver before it embarks on transmission which can add some latency overhead.

Hybrid rate-limiting approaches can be used to support a large number of priority classes while reducing hardware complexity and keeping scheduling precision high. NicPic [129] classifies and stores packets in queues located in RAM and labels them with proper rate limits by host CPU. Packets are then fetched by NIC via DMA and scheduled using hardware rate-limiters. NIC first decides on which flow's packets to send and then pulls them from RAM.

As the last option, rate-limits can also be applied at the application layer. Applications can do this by limiting the volume of data handed off to transport layer over periods of time. This approach is simple but requires changes to applications. Also, rate-limiting precision will be limited. Finally, this may lead to bursty traffic as incoming application data to transport layer may get buffered prior to transmission on the wire, i.e., applications have no control over how data is eventually sent.

TABLE VII
OVERVIEW OF TRAFFIC SHAPING TECHNIQUES

Scheme	Description	Limitations
Rate Limiting §V-B1	Limits the transmission rate of outgoing packets from the rate limiter. A token bucket is usually used to limit maximum persistent packet transmission rate to token arrival rate.	Rate limiting has limited accuracy depending on how it is enforced, in software (usually less accurate but cheaper) or hardware (usually more accurate but needs hardware support). Also, various implementation techniques in software lead to different accuracy such as in the operating system kernel or using kernel bypass modules (see §VII-D).
Packet Pacing §V-B2	Inserts time spacing between consecutive packs to spread them uniformly across a window of round trip time (RTT). This reduces traffic burstiness and average network buffer occupancy, therefore improving end-to-end latency.	Packet pacing has limited accuracy and similar to rate-limiting, its accuracy depends on the implementation technique used.

2) Packet Pacing: Packet Pacing is the process of adding space between consecutive packets so that they do not arrive at the network back to back which reduces traffic burstiness. Burstiness can degrade network performance in several ways. Long bursts can overload switch buffer and create consecutive packet drops. Average latency of all packets then increases since they have to wait in longer queues. In addition, it creates transmission rate oscillations making it hard to do careful bandwidth allocation [9], [29].

Earlier work [124] experimenting with pacing in a general network setting has shown that it can considerably reduce queuing delay. Combined with other types of congestion signals, pacing can improve the performance by evenly distributing traffic across the timeline [9]. In addition, pacing should only be applied to long-running and throughput-oriented flows to reduce their impact on short latency-sensitive flows [9]. The benefit of pacing depends on the network bandwidth-delay product, buffer size, and the number of flows. Having so many flows that share a buffer can reduce the effectiveness of pacing due to inter-flow burstiness and creation of synchronized drops [125].

Pacing can be done in both hardware and software, but hardware pacing can be more effective due to higher scheduling precision, especially at high rates where spacing between packets is tiny [9]. Software pacing may be performed by end-hosts as part of a driver or a kernel module. In cloud environments, due to widespread use of virtualization, packets may be paced at the virtual interfaces in the hypervisor software. Pacing in software may be overridden by NIC offloading features such as LSO, if enabled, since NIC is the actual entity that sends packets out on the wire. In hardware pacing, packets are buffered at the NIC, each assigned a timer, and scheduled to be sent when their timer goes off.

Pacing can also be done at the network edges (e.g., ToR switches) as opposed to end-hosts. For example, Queue Length Based Pacing (QLBP) [142] uses a pacing controller attached to edge queues to determine when the next packet in the queue is supposed to be sent as a function of queue length.

C. Prioritization

Table VIII provides an overview of this section. Classifying flows based on their priorities and treating them accordingly can improve performance. Such prioritization can be done in-network by using multiple queues at the switches and allowing higher priority traffic to go over lower priority traffic [53], [122], [134], [135], and at the senders by performing rate-control according to priorities [28], [51], [86], [119].

Priorities are usually assigned either based on flow size to minimize mean latency (by mimicking SRPT scheduling policy) [53], [134] or based on deadlines to minimize the number of deadline missing flows [80], [122]. Control traffic is naturally prioritized to improve the feedback timeliness and quality (e.g., ACKs in TIMELY [99] and Trimmed Packets in NDP [103]) or decrease control loop delay (e.g., RTS in pHost [80]).

For many applications, flow sizes are either known or can be roughly estimated upon initiation [51], [53], [86], [122] making it easy to assign priorities by size to reduce mean latency. In case flow sizes are unknown apriori, dynamic prioritization can be used where packets of a flow first receive the highest priority, but get demoted to lower priorities as more of them is seen.

For example, dynamic Packet Prioritization (DPP) [133] uses two queues, an express queue which has higher priority and a normal queue. It reduces the priority of long running flows by counting their packets against a threshold. Having multiple queues allows for more precise classification of flows [122], [134]. However, recent work shows that most benefit in reducing mean FCT can be obtained using up to 8 queues [53]. Finding proper threshold values based on which flows are demoted to lower priorities may require solving an optimization problem that takes into account the flow arrival rates and flow size distribution. In datacenter environments with known traffic characteristics and workloads, such thresholds may be determined offline and applied in real-time [134]. It is also possible to virtually emulate the behavior of having infinite number of queues using only two actual queues per switch port, a high and a low priority queue [143]. This can be achieved by assigning flows with highest priority to high priority queue while the rest of flows to low priority queue and dynamically changing flows assigned to high priority queue when other flows with a higher priority complete.

Prioritization can be performed fully at switches by keeping state on flows passing through and using some priority assignment criteria such as the total number of packets sent. This simplifies end-hosts at the expense of higher computational and memory burden on the switches.

TABLE VIII
OVERVIEW OF PRIORITIZATION TECHNIQUES

Property	Scheme	Description (Benefits/Drawbacks)
Classification	Static	A flow's priority is fixed once assigned. This approach can be applied when flow properties (e.g. size) are known apriori.
	Dynamic	A flow's priority may change over time according to its behavior, i.e., number of packets sent over time. This approach can be used if flow properties are unknown apriori.
Criteria	by flow size	Mimics the Shortest Remaining Processing Time (SRPT) scheduling discipline which aims at minimizing mean flow completion times.
	by flow deadline	To minimize deadline miss rate or lateness by first satisfying flows with closer deadlines.
	by class of service	If an application or a tenant has a higher service level requirement or agreement, flows associated with them can be prioritized accordingly to minimize the effect of other applications/tenants using the same physical network.
Location	at the switches	Switches can keep state information on flows passing through them and determine their priority according to flows' behavior. For example, in case of prioritization by flow size, switches can estimate flow sizes by counting the number of packets they have sent (mimicking least attained service discipline). Keeping state information at the switches may be costly when there are many flows.
	at the end-hosts	End-hosts can mark packets with priority tags allowing switches to simply enforce priorities according to tags. This reduces switch complexity but requires changes to the end-hosts' software or hardware protocol stack.
Implementation	at Layer 2	Ethernet standard IEEE 802.1Q priority based forwarding.
	at Layer 3	Differentiated Services (DiffServ) can be used at the IP layer.
	Custom approaches	Can be used by adding support to switches and/or end-hosts, may require changes in software/hardware to switches and/or end-hosts.
Enforcement	Strict	A lower priority flow is only sent when there are no packets available from any of the higher priority flows. This minimizes the effect of lower priority flows on higher priority ones but can lead to starvation of lower priority flows.
	Non-strict	A lower priority flow can be sent even if there are packets available from higher priority flows. This occurs when a required volume of higher priority flows' demand is satisfied (e.g. one low priority packet is sent for every $K \geq 1$ high priority packets sent) and mitigates the starvation problem of lower priority flows.

Another approach is for the end-hosts to tag flows with priority tags while switches just process tags and put packets in proper queues [134], [135]. End-host priority tagging can be done at the NIC, OS kernel, hypervisor, or even by applications before packets are sent to the network. In case of virtualization, if end-host VMs cannot be trusted to assign priorities properly, middleboxes can be used (e.g., at the hypervisor) that monitor and tag packets (e.g., using OpenVSwitch [144]) which applies to both static [53] and dynamic [122], [134] prioritization.

Priorities can also be assigned in different ways. Ethernet standard IEEE 802.1Q priority based forwarding [145] that provides 8 levels is supported by many switch vendors and can also be used in datacenters [135]. At the IP layer, Differentiated services (DiffServ) [146] can be used [147]. Custom queuing techniques and headers can also be used which may require changes to both switches and end-hosts [53].

Strictly prioritizing flows can lead to starvation where lower priority flows cannot make progress due to large volume of higher priority traffic. A simple solution is to use weighted queuing instead of strict prioritization. For instance, DAQ [121] uses a weighted round-robin between long and short flows to make sure that throughput-oriented flows keep making progress. Aging can also be used to address starvation while minimally affecting critical flows. An aging-rate can be used to increase the priority of low priority flows as a function of their waiting time [86].

D. Load Balancing

Datacenter topologies typically provide a large degree of path redundancy. Properly distributing load across these paths reduces contention among flows while increasing overall resource utilization. Without effective load balancing many links may not be utilized while some experiencing congestion [48], [148]. Table IX provides an overview of general load balancing concepts and their tradeoffs.

Load balancing can be static or dynamic (adaptive). Static approaches use a fixed criteria to assign traffic to available paths such as by hashing specific fields from packet headers. For example, ECMP [150] is a popular static load balancing technique that only distributes load across equal cost paths. Adaptive load balancing dynamically selects paths for traffic according to distribution of load to minimize hot-spots. Various criteria can be used for path assignment such as per-hop or per-path queue occupancies [151]. After choosing the initial path according to current load, some adaptive approaches keep monitoring the network status and distribution of traffic. They change direction of traffic to eliminate or reduce hot-spots. These approaches are referred to as reactive. If not applied with care, reactive load balancing might lead to oscillations.

Examples of reactive dynamic load balancing techniques include Planck [152], Hedera [116], MPTCP [45], DIBS [83] and CONGA [105]. Planck uses a controller that monitors traffic and generates congestion events that include the

TABLE IX
OVERVIEW OF LOAD BALANCING TECHNIQUES

Property	Scheme	Description (Benefits/Drawbacks)
Classification	Static	A new flow is assigned to any of the available paths using some fixed criteria such as by hashing parts of its packets' header. This approach is simple but inflexible. For example, in case two throughput oriented flows are assigned to the same path, they cannot be moved to other less utilized paths later.
	Dynamic (reactive)	Flows can be moved across any of the available paths according to available bandwidth. Offers a better performance in general but adds the complexity of measuring link utilizations, accounting for flows, and calculating best flow assignments accordingly.
	Dynamic (proactive)	After a flow is assigned to one of the available paths according to some criteria, its assignment will remain fixed. The initial assignment is performed according to network conditions such as available bandwidth. This approach is somewhat between the previous two assignments above in terms of implementation overhead, flexibility and performance.
Granularity	per packet	Finest load balancing but can lead to high reordering.
	per flow	Coarse load balancing but achieves minimal reordering.
	per flowlet	A flowlet's size dynamically changes according to differences of latencies of candidate paths. At high rates and/or high latency difference between available paths, a flowlet can become significantly large. As a result, this can result in both fine and coarse grained load balancing (it is always somewhere between per packet and per flow). Flowlets have been found effective for load balancing over asymmetric (i.e., with different available bandwidth) paths [149]. As a drawback, flowlets may cause reordering of small flows and hurt their completion times.
	per flowcell	A flowcell has a fixed size that is usually about tens of packets. Using flowcells simplifies load balancing compared to flowlets (no need to carefully measure path latencies and schedule accordingly) and reduces possible reordering of small flows. It can however significantly increase reordering for larger flows that will be broken into many flowcells.

transmission rate of flows passing through the congested link. It then routes traffic away from congested spots. Hedera initially places flows via hashing, and then uses a central controller to monitor the network, detect long running flows and reschedule such flows on a lightly loaded path to balance the load. MPTCP establishes multiple sub-flows from the beginning and then shifts load between them according to the level of congestion observed across each sub-flow. DIBS forwards packets that arrive at a full queue to one of the nearby switches instead of dropping them which will be forwarded towards the destination through another path. CONGA proposes a technique for leaf-spine topologies [13] based on lazy evaluation. A leaf switch has a table which holds the load seen along its outgoing paths. Such load information is collected by receiving switches and then piggybacked on traffic.

Some proactive adaptive approaches include DeTail [30], Presto [98] and Expeditus [106]. DeTail uses a per-hop adaptive method and operates in lossless environments with layer 2 flow control [153]–[155]. At every hop, packets are forwarded to the egress port with minimal queuing. Presto breaks flows into small units called cells and sends them across all available paths. This implies that small flows are kept intact as long as they fit into one cell. Expeditus dynamically assigns flows to paths in 3-tier Clos topologies. It uses dedicated packet tags to communicate load information across switches for path selection upon arrival of a new flow. For path election, the upstream switch sends a message to its downstream peer expressing congestion at its egress ports. The receiving switch compares the upstream congestion metrics with the ones for its ingress ports choosing the ports that minimize the maximum congestion along the path.

Load balancing can be done per-packet, per-group of packets that belong to the same flow, and per flow. While considering these options, two important performance metrics are packet reordering and distribution of load across the network. Reordering is known to waste server resources and increase latencies [67].

per flow load balancing minimizes packet re-ordering. Congested spots might be created in case multiple large flows are assigned to the same links. Even when carefully placing flows, per flow load balancing provides limited benefit if there is a large number of throughput-oriented flows [151]. To improve performance, one might reroute flows from their initially assigned paths according to network conditions. However, moving flows from their initial paths might still result in re-ordering. For example, Hermes [156] performs per flow load balancing added that it can perform fast rerouting of flows according to network conditions. It can potentially reroute flows per packet in case conditions change rapidly and continuously. One however needs to consider the stability of such schemes, especially as load increases, since many flows may be interacting in the network (in case of instability, such schemes may keep changing routes, which increases reordering, while not improving load balancing). Flier [157] is another flow-level load balancing scheme that reroutes flows according to level of congestion observed (via checking ECN markings) and failures (by paying attention to timeouts).

Per-packet load balancing provides the finest balancing degree but leads to packet re-ordering. For every new packet, one of the available paths can be chosen either randomly, according to usage history, or adaptively based on the distribution of load. Valiant Load Balancing (VLB) [158] can uniformly distribute packets across paths. Packet Spraying [159] (a.k.a. packet scatter) uses a Round Robin approach to multiplex packets of a flow across all possible next hops. Packet scatter can greatly balance load at the network core [80], [139]

and reduce latency [53], [160]. Another per-packet approach, DRB [127], reduces network queuing and buffering required at the receiver by choosing the forwarding paths that avoid clustering of packets between same source-destination pairs. DRILL [161] determines the forwarding path of every packet of a flow independently by considering per port local queuing at the switches sending an arriving packet to the least occupied queue. It is argued that DRILL's packet reordering is minimal due to similar latencies across all the paths between every source and destination pair due to well-balanced switch queue occupancies.

Another option is to group several packets of a flow and perform load balancing *per-group of packets*. Packets can be grouped according to their inter-packet delay or their accumulative size. In the former case, all packets of a flow whose inter-packet delay is less than some timeout threshold form a flowlet [126]. Flowlet scheduling essentially makes use of natural traffic burstiness for load balancing Section III-C3. In the latter case, packets are grouped with a limit on total group volume to form flowcells [98], [128]. Each flowlet or flowcell can be sent over a different path to distribute load.

In flowlet scheduling, smaller timeout values allow for finer load balancing while larger values reduce reordering. To minimize reordering, one can choose the timeout value to be greater than the difference between latencies of paths with minimum and maximum latencies. Flowlets have been found to effectively balance load while incurring minimal reordering in datacenters [105], [113]. However, dependence of flowlet switching on inter-packet intervals could lead to creation of arbitrarily large flowlets at high rates, which could lead to congestion in case they collide. Another drawback is the possibility that small flows are split into several flowlets which could lead to reordering and increased latency.

In flowcell scheduling, smaller flowcells balance load better while larger ones reduce reordering. Flows shorter than the cell size are guaranteed to be sent on a single path minimizing their latency and reordering. Previous work has used grouping thresholds of tens of kilobytes (10 KB at ToR switches [128] and 64 KB at the hypervisor layer [98]) to effectively spread the load across the network and reduce creation of random hot-spots. As a drawback, this approach may lead to higher reordering for long flows compared to flowlets.

1) Data and Task Placement: Many datacenter applications need to access data stored on multiple servers to perform computations or respond to queries. Therefore, placement of data determines what options are available to access them. Such data could be a value for a key in a distributed key-value store or an object in a replicated or erasure coded store. For example, any of the replicas in a replicated store can be accessed or any of the k pieces out of n pieces of data would allow data recovery in an $(n,\ k)$ erasure coded storage system.

Pieces of data can be distributed across racks and servers (depending on topology) to allow wider load balancing options. For example, Ceph [162] uses Selective Replication that distributes copies of the original data across the cluster according to their popularity. Ceph also distributes contents of large directories and ones with lots of writes across many servers to reduce hot-spots. HDFS [163] allows for similar features but also considers the network topology while distributing replicas. For example, there is better connectivity and usually higher available bandwidth within a rack.

Placement of tasks (execution) could be as important as placement of data. Task schedulers and resource managers can place computation in accordance with placement of data to reduce network usage, contention for network access and queuing [164]–[169]. A task scheduler may consider the flow scheduling policy of the network (FCFS, SRPT, Fair Sharing, etc.) in addition to placement of data to improve overall task completion times [170].

2) Routing and Forwarding: Switches forward packets according to Forwarding Information Base (FIB) which contains a set of rules that determine outgoing port(s) for incoming packets. FIB rules can be installed proactively or reactively. Proactive installation may result in a larger number of rules as not all of them may be used at all times while reactive installation of rules may incur setup time overhead. Such rules can be installed either directly or by a routing protocol that calculates the rules and installs them such as BGP, IS-IS or OSPF. In case of routers, FIB usually reflects a subset of Routing Information Base (RIB) which is a table of routes learned or calculated by a router. For load balancing, various forwarding techniques can be used to direct traffic across several paths.

Standard distributed protocols can be used for load balancing. As a Layer 2 solution, VLAN based load balancing puts same machines on several virtual networks allowing traffic to be spread across paths via using different VLAN tags. It however provides limited scalability due to creation of large broadcast domains. Layer 3 routing for large networks with support for load balancing can be used in case multiple next hops are available for a destination. For example, Equal Cost Multipathing (ECMP) statically selects the next hop by hashing packet header fields. For fast convergence, IGP routing protocols can be used such as IS-IS or OSPF. Load balancing using these protocols is challenging since path costs need to be exactly equal and the number of supported equal paths is limited. BGP provides higher flexibility for this purpose [171] and can be customized to converge fast [172].

Load balancing can be performed using centralized techniques. In Layer 2, scalable forwarding can be built by replacing the default MAC address broadcast and discovery approach with a centralized one [173], [174]. A controller can then setup Layer 2 forwarding that accounts for path diversity [174]. Centrally implemented Layer 3 approaches allow FIBs to be calculated centrally using free routing software stacks such as Quagga [175], and installed on switches to implement various protocol stacks such as OSPF and IS-IS with ECMP support which provides much higher flexibility [5], [43], [176], [177]. For example, to help BGP converge faster, a central controller can calculate and update BGP tables in routers to achieve desired forwarding behavior [4]. Simpler centralized approaches in which new FIB rules are installed directly by a controller upon arrival of a new flow can also be used. Forwarding can be done in a way that distributes load, either by hashing or adaptively selecting least loaded paths. Using this approach, careful consideration of scalability is necessary.

Previous approaches relied mostly on network for routing. An end-host based approach is Source Routing which simplifies the network by moving the forwarding information to packets, and eliminating the need to disseminate updates of forwarding state [178]. To properly encode paths in packets, end-hosts need to be aware of network topology and paths. In addition, a mechanism is needed to detect and disseminate network status and failure information to the end-hosts. BCube [41] uses probe packets to measure available bandwidth across paths and assign new flows to least loaded ones. PSSR [136] encodes a list of outgoing port numbers instead of addresses for next hops to decouple addressing from forwarding.

3) Effect of Failures: The scale of datacenter networks has made failures an important concept. Even using high quality and expensive equipment, the probability that some network element fails (e.g., a switch, a port or a link, etc.) can be considerably large at any moment [19]. When some network equipment fails, capacity in the network decreases accordingly; however, since datacenter networks are usually connected with large degrees of redundancy, network failures rarely lead to complete inability to reach parts of the network. However, the capacity loss due to failures in some parts of network can lead to complications in load balancing by affecting the effective capacity of different paths, i.e., creating capacity asymmetries. This may not have a significant impact on topologies that are inherently asymmetrical (e.g., JellyFish, Xpander, etc.); however, more basic load balancing techniques, such as ECMP, which are used in symmetric topologies (e.g., Fat-Tree, Leaf-Spine, VL2, etc.), will have a hard time effectively distributing load in case of failures [18], [105], [149]. This is noteworthy considering that many industry datacenters are based on symmetric topologies.

A variety of solutions have been proposed to perform better load balancing in case of capacity asymmetries across paths examples of which include WCMP [18], CONGA [105], HULA [179], Presto [98], LetFlow [149], DRILL [161] and Hermes [156]. WCMP mitigates asymmetries by extending ECMP and assigning weights to different paths proportional to their capacity referred to as weighted traffic hashing. Using WCMP, weights determine the number of hash entries per outgoing port at the switch which are selected proportional to capacity. CONGA and HULA operate by performing pathwise congestion monitoring and shifting traffic accordingly. As a result, if capacity of a path is reduced due to failures, its congestion metric will increase faster and it will automatically be assigned less traffic. Presto applies a weighted forwarding mechanism in a way similar to WCMP where weights are pushed to the end-host virtual switches. LetFlow is a simple approach where flowlets are used as means to dynamically adapt to varying path capacities. LetFlow relies on natural property of flowlets which allows them to shrink or expand (in size) according to available capacity over paths. DRILL performs load balancing over asymmetric topologies by first breaking them into groups of symmetric components and then performing load balancing on them per switch. Symmetric paths should have equal number of hops and their queues should be shared by same flows at every hop from source to destination. Hermes uses comprehensive sensing at the end-hosts to monitor path conditions and reroute flows affected by failures or congestion caused by asymmetries. Hermes considers monitoring of latency and ECN markings for detection of congestion while looking at frequent timeouts and retransmissions as signs of failures. In addition, to improve visibility into path conditions, Hermes uses active probing by sending small probe packets between end-host pairs periodically.

E. Multipathing

To improve the overall throughput for a single flow and provide better reliability in case of failures, multi-path techniques can be used [45], [68], [139], [180]. A flow is split into multiple sub-flows each sent over a different path. To receive traffic on multiple paths, the receiver needs to buffer data received from each sub-flow and put them in order. Buffering is proportional to sum of throughput across paths times the latency of the longest path [68], [181]. Although latencies are quite small in datacenters, link bandwidths can be significantly high. Additional sub-flows can generally increase both memory and CPU utilization [68].

Depending on flow sizes, the applications may decide whether to use multiple sub-flows. Overhead of setup and teardown for multiple sub-flows may be considerable for short flows. For long-running background flows, using every bit of bandwidth available through multipathing may improve their total average throughput.

Several examples of multipath transports include MPTCP [139], XMP [182] and MMPTCP [183]. MPTCP leverages ECMP to route sub-flows over various paths and increase total throughput while balancing load across paths by moving load across sub-flows. XMP approximates the solution to an optimization problem that maximizes utilization. It uses RED [184], [185] with ECN marking to keep the queue occupancies low. XMP achieves most benefit by using two sub-flows. MMPTCP aims to improve FCT for short flows by employing a two phase approach. First phase uses packet scatter by randomizing source port numbers and routing via ECMP to minimize FCT for short flows and second phase uses MPTCP to maximize throughput for long flows.

Although multipathing is generally helpful in increasing utilization, the benefit it offers is limited under various conditions. Under heavy workloads, multipathing may not improve throughput if most paths are already highly utilized [86]. In addition, if paths have different characteristics, such as latency or capacity, multipathing may offer marginal increase (or even decrease) in throughput [180]. This may occur in datacenters in case different communication technologies are used across machines, such as a combination of wireless and wired networks or if available paths have varying number of hops or different capacities. Another complication is caused by overlapping paths where a multihomed sender's paths to the receiver actually have common edges [186]. This can occur in datacenters as well in case of link failures which reduce available paths or if sub-flows are mistakenly hashed to the the same links. Finally, multipathing may lead to unfairness if multipath flows share a bottleneck with regular flows [187].

TABLE X
SUMMARY OF SCHEDULING TECHNIQUES

Scheme	Description	Limitations
Reservation §V-F1	Reserving bandwidth prior to a sender's transmission minimizes congestion and queuing latency.	Reservation adds the latency overhead of calculating a transmission schedule before a flow is allowed to transmit. In addition, it is challenging to enforce a transmission schedule network wide. Inaccuracies in transmission rates may necessitate continues updates to the schedule which is costly.
Redundancy §V-F2	A flow can be replicated multiple times to reduce the effect of high tail latency. The fastest reply is obtained and then replicas are terminated.	Effective if there is a considerable gap between tail and median latency. In addition, it is less effective when network is heavily loaded.
Deadline-Awareness §V-F3	Scheduling can be done according to deadlines to minimize deadline miss rate and lateness.	It may not be possible to meet all the deadlines in which case it should be determined whether deadline miss rate is more important (hard deadlines) or lateness (soft deadlines). In presence of deadlines, it is unclear how to effectively schedule traffic if flow sizes are not known apriori or cannot be estimated.
Disciplines §V-F4	A variety of scheduling disciplines can be applied according to desired traffic control objectives. For example, SRPT minimizes mean latency while Fair Queuing maximizes fairness.	Disciplines can usually optimize for only one performance metric. A mix of scheduling policies can hierarchically optimize for more than one objective. If a utility of objectives is desired, well-known policies may provide solutions far from optimal.
Preemption §V-F5	Allows the network to update current schedule (along with already scheduled flows) according to new flow arrivals.	Preemption may offer limited benefit if all flows have similar properties (i.e., size, deadline, etc.).
Jittering §V-F6	Prevents a sender from initiating many flows together. For example, this helps mitigate the incast problem §III-C6.	This approach only offers very coarse grained control over incoming traffic.
ACK Control §V-F7	By carefully controlling when ACKs are sent to senders, a receiver can control the incoming flow of traffic.	This approach only offers coarse grained control over incoming traffic.

Unfairness may also arise when multipath flows with different number of sub-flows compete for bandwidth over a bottleneck.

F. Scheduling

Given a list of flows with their priorities and demands, the scheduling problem aims to optimize an utility function of several performance metrics such as utilization, fairness or latency. The objective is usually to maximize throughput for bandwidth-hungry flows and minimize FCT for latency-sensitive flows considering fairness among flows in each class of service. Different scheduling techniques can be used to reduce FCT, provide better bandwidth guarantees to long-running flows, and help deadline flows meet their deadlines. In general, scheduling a mix of flow types requires formulation of a complex optimization problem that is generally computationally expensive to solve. Table X offers an overview of scheduling techniques presented here.

1) Reservation: To provide better bandwidth guarantees and prevent creation of congestion spots resources may be first checked for availability and then allocated before an end-host can start transmitting a flow. Requesting resources can be done in different units and such requests might be processed centrally or in a distributed fashion.

In a fully centralized approach, end-hosts can report their demands to a central scheduler and ask for transmission slots or rates. Some examples are TDMA [111], FastPass [112], FlowTune [113] and TAPS [188]. TDMA uses a coarse-grained centralized approach in which end-hosts send their demands to a fabric manager which then allocates them contention-less transmission slots. The scheduling is performed in a round by round basis where each round is several slots during which different hosts can communicate over the fabrics. FastPass allocates slices of time on a per-packet basis to improve utilization and considers variations in packet size. FlowTune performs centralized rate allocation on a per-flowlet basis. TAPS uses a central SDN controller (please refer to Section VII-A) to receive and process flow requests with known demands, verify whether deadlines can be met on a per-task basis, allocate contention-less slices of time for senders to transmit and install forwarding rules.

Distributed reservation can be done upon connection setup. A sender can request the rate at which it would like to transmit. This rate along with the available bandwidth is considered by network fabrics to determine the rate and path of the flow. Allocated rate can be piggybacked on ACKs. RCP [85] and PDQ [86] perform the rate allocation at switches with this approach. Receiver-based reservation can also be used. Receivers view multiple incoming flows and determine their transmission schedule. Senders can also communicate their demands to the receiver which calculates the transmission schedule [189]. In addition, token-based techniques can be used where a receiver sends back tokens to allow senders to transmit a unit of data (e.g., packet) per token. The token-based approach has been found very effective in addressing the incast problem which is achieved by controlling the rate at which tokens are sent back to senders from the receiver to ensure that data arriving at the receiver conforms with available capacity [80], [103], [104].

2) Redundancy: Tail latency is an important quality metric for many latency-sensitive applications, such as search, as

it determines quality of user experience. Late flows can take much longer than median latency to finish [29], [62]. An effective approach is to replicate flows, use the fastest responding replica and terminate the rest. For simplicity, creation of replicates may be performed completely at the application layer. The probability of more than one replica being late is usually significantly small. Replicated flows can be scheduled on different paths and can even target different servers to reduce correlation.

One approach is to replicate every flow and then take the one whose handshaking is finished earlier and terminate the rest [190]. Another approach is to only replicate slow requests by reissuing them [62]. It is necessary to judiciously decide on the number of redundant requests to balance resource usage and response time. Since only a tiny portion of all flows are usually laggards, additional resources needed to allow large improvements may be small [62].

3) Deadline-Awareness: For many applications, criticality of flows can be captured as deadlines. Scheduling techniques are expected to minimize deadline miss rate. In case of hard deadlines, in which case delivery after deadline is pointless, flows can be terminated early if their deadlines cannot be met. D2TCP [28], D3 [51], PDQ [86], and MCP [120] are examples of deadline-aware scheduling schemes that reduce deadline miss rate by performing rate control according to flow deadlines, either by explicit rate allocation (D3, PDQ) or implicitly by adapting sender's outstanding window size (D2TCP, MCP). Tempus [59] formulates a complex optimization problem to maximize the fraction of flows completed prior to their deadlines while considering fairness among flows. Amoeba [60] aims to guarantee deadlines by performing initial admission control via formulating an optimization problem. RCD [191] and DCRoute [192] aim to quickly determine whether deadlines can be met by performing close to deadline scheduling and guarantee deadlines via bandwidth reservation. In addition, considering task dependencies in meeting deadlines can help reduce deadline miss rate of tasks [188], [193].

While it is important to meet deadlines, finishing deadline flows earlier than necessary can hurt the FCT of latency-sensitive traffic [56]. In general, we can form an optimization problem for scheduling flows [120]. A study of how well-known scheduling policies perform under mix flow scenarios with varying fraction of deadline traffic can be found in [194].

4) Disciplines: The following are some well-known scheduling disciplines. First Come First Serve (FCFS) is a simple policy where a task has to be completed before the next task can begin. FCFS provides bounded lateness when scheduling flows with deadlines [195] and is close to optimal for minimizing tail completion times given light-tailed flow size distributions [196]. Processor Sharing (PS) divides available resources equally among flows by giving them access to resources in tiny time scales. Fair Queuing (FQ) approximates PS within transmission time of a packet and can be used to enforce max-min fairness. Earliest Deadline First (EDF) minimizes deadline miss rate for deadline flows. Shortest Job First (SJF) minimizes mean flow completion time in offline systems where all flows and their demands are known apriori. For online systems where requests can be submitted at any time, Shortest Remaining Processing Time (SRPT) which is preemptive, minimizes mean FCT [197]. SRPT also offers close to optimal tail completion times while scheduling flows with heavy-tailed size distributions [196]. Least Attained Service (LAS) [198], which prioritizes less demanding flows, can be used to approximate SJF without apriori knowledge of flow demands [199].

Many works use policies that approximate or implement well-known scheduling disciplines. RCP [85] performs explicit rate control to enforce processor sharing across flows sharing the same links. PDQ [86] combines EDF and SJF giving higher priority to EDF to first minimize deadline miss rate and then minimize FCT as much as possible. FastPass [112], implements *least recently allocated first* giving each user at least as much as their fair share to reach global user-level max-min fairness. Also, across flows of a user, *fewest remaining MTUs first* is used to minimize FCT by emulating SJF. pFabric [53] and SFS [79] follow the shortest remaining flow first which is essentially SRPT. PIAS [134] uses a dynamic priority assignment approach which approximates LAS by counting the number of packets sent by flows so far. RACS [87] uses weighted processor sharing with configurable weights to approximate a spectrum of scheduling disciplines such as SRPT and LAS.

Aside from research, one may be interested in what disciplines can be enforced using industry solutions. Switches by default provide support for FIFO queues per outgoing port which enforce FCFS scheduling policy. Some switches provide support for multiple levels of priority at the outgoing ports (multiple FIFO queues with different priorities). Using these queues, it is possible to mimic the SRPT scheduling policy by putting smaller flows into higher priority queues. For example, Cisco offers switches that support this feature using dynamic packet prioritization (DPP) [133] which tracks flows as their packets arrive (estimating a flow's size according to LAS policy) and assigns them to priority queues according to their sizes. Weighted Fair Queuing (WFQ) is also supported by many switch vendors per "class of service" or per flow where weights determine the proportional importance of flows, i.e., a flow is assigned bandwidth proportional to its weight.

5) Preemption: Many practical systems have to address arrival of requests in an online manner where requests can arrive at any time and have to be addressed upon arrival. Order of arrivals can impact the performance of scheduling algorithms due to race conditions which can lead to priority inversion. For example, in an online scenario with the objective of minimizing mean FCT, SJF might perform poorly if many short flows arrive shortly after a large flow. Preemptive scheduling policies (SRPT in this case) can be used to address this problem [86].

6) Jittering: A server issuing many fetch requests can use jittering to desynchronize arrival of traffic from various flows and reduce peaks in traffic volume. Such peaks can lead to temporary congestion, dropped packets and increased latency. Jittering can be applied by adding random delays at the application layer when initiating multiple requests at the same time [29].

TABLE XI
SUMMARY OF OPEN CHALLENGES

Open Challenge	Description
Handling Mix Workloads	Datacenter environments house a variety of applications that generate flows with different properties and requirements. Effectively handling the mix of traffic workload requires clearly defining a utility function of performance metric variables (delay, utilization, deadline miss rate, lateness, fairness) and formulating an optimization problem.
Load Balancing vs. Packet Reordering	To minimize packet reordering, a sender needs to carefully schedule packets over all available paths while paying attention to other traffic workload. Enforcing a no reordering constraint can result in lower network utilization. As a result, to increase utilization while imposing acceptable level of reordering, one can consider a utility function of these factors and formulate an optimization problem solving which provides a desirable transmission schedule.
Achieving High Throughput and Low Latency	In distributed traffic control approaches, a feedback from network is provided to senders to adapt their transmission rate. While transmitting at high rates, the network may not provide feedbacks fast enough leading to network overload and congestion. Therefore, limited network responsiveness may increase average queuing delay and latency at high throughput.
Objective Mismatch	Due to resource constraints, it may not be possible to come up with a transmission schedule where all performance objectives are optimal. It then becomes necessary to define a utility of performance metric variables and aim to maximize utility by formulating an optimization scenario.

7) ACK Control: ACKs may be used as part of the network traffic scheduling process since they determine how a sender advances its outstanding window of bytes. They can be thought of as permits for the sender to transmit more data. A receiver can stall a sender by intentionally holding back on ACKs or limit sender's rate by delaying them. For example, a receiver can pause ACKs for low priority flows upon arrival of higher priority traffic and can generate ACKs to control which flows are assigned more bandwidth [79]. In addition, by reporting a small receiver window in ACKs, a receiver may limit the transmission rate of a sender [65], [78]. This approach has considerable similarities with the token-based transmission control applied in schemes such as pHost [80], NDP [103] and ExpressPass [104].

VI. OPEN CHALLENGES

In this section, we point to a few open challenges with regards to traffic control. To find an optimal solution, these problems may be modeled as complex optimization scenarios that are computationally expensive to solve (large number of variables and constraints, presence of integer variables and/or non-linear constraints, complex objective functions) and practically hard to enforce (lack of hardware support, slow response time of software implementations, presence of failures and errors). Current approaches apply a variety of heuristics and simplifying assumptions to come up with solutions that are practical and attractive to industry. In addition, such optimization scenarios may be infeasible due to presence of contradictory constraints meaning it may not be possible to optimize for all objectives given available resources. Therefore, it becomes necessary to relax some requirements. In the following, we do not provide any optimization models, rather we point to cases where such complex models may appear. Table XI offers an overview of open challenges in this section.

Handling Mix Workloads: Datacenter environments are usually shared by a variety of applications that generate different network workloads with different performance metrics. For example, two applications of search and backup may be sharing the same network; while search network traffic requires low communication latency, backup network traffic demands high throughput. Some network workloads may have deadlines, either soft or hard, which should be considered along with non-deadline network traffic. Effectively scheduling such mix of network workload is an open problem. As a solution, one can formulate a complex optimization problem for such scheduling that considers maximizing some utility function. This utility could be a function of performance metric variables such as deadline miss rate, average lateness, link utilizations, and latency which represents the value achieved by tenants and operators. For example, one can examine how the objective presented in [56], i.e., to reduce per packet latency, translates into utility.

Load Balancing vs. Packet Reordering: In datacenters, there is usually a large number of paths between any two end-hosts which according to topology, could have equal or unequal lengths (number of hops and latency). To use all available paths, one could apply load balancing techniques. Per packet load balancing can provide the finest level of balancing but in general leads to significant packet reordering. In contrast, although per flow load balancing does not lead to packet reordering, it may not be effective in using available bandwidth over many parallel paths. In general, effective load balancing necessitates scheduling of packets over available paths according to available bandwidth. One could consider an additional constraint that allows sending packets on a path only if such scheduling does not lead to out of order arrival of packets at the receiver. But this might reduce utilization by decreasing packet transmission opportunities. As a result, effectively utilizing available bandwidth over all paths could be at odds with minimizing packet reordering. One could relax the no-reordering constraints by allowing reordering to some extent. This relationship can be formulated as a utility of reordering and bandwidth utilization maximizing which in general is an open problem. For example, the works discussed in Sections V-D and V-E offer several possible approaches.

Achieving High Throughput and Low Latency: There is a strong connection between traffic control and the control theory. A traffic control scheme depends on concepts from control theory in managing flow of traffic across the network: transmission rate of senders is a function of feedback received

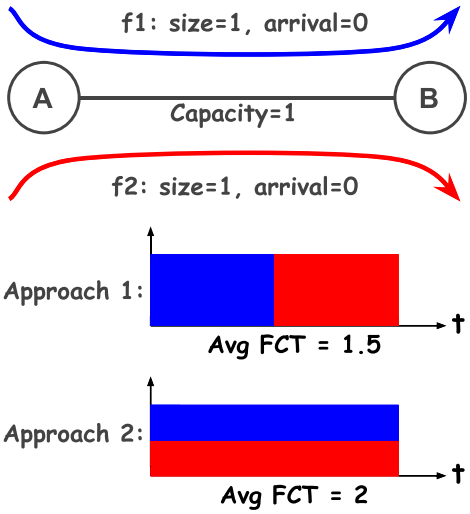


Fig. 5. It may not be possible to optimize all performance metrics together (mismatch between fairness and mean FCT).

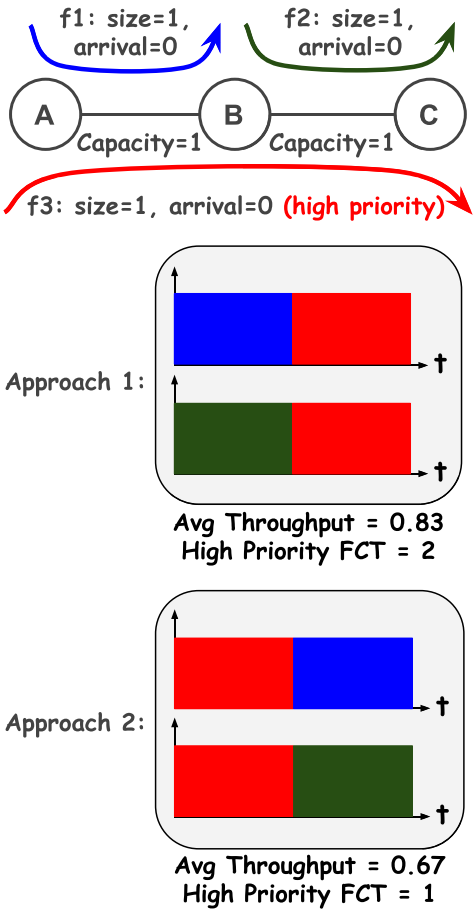


Fig. 6. It may not be possible to optimize all performance metrics together (mismatch between average throughput of transfers and FCT of high priority traffic).

from the network, and the time it takes from when a sender changes its rate until it receives a feedback of that change constitutes the loop delay. Due to existence of this loop delay, a network has limited responsiveness due to processing and propagation latency as well as queuing delay. The former two factors are usually much smaller in datacenters and queuing delay determines responsiveness. When transmitting at high rates, it is easy for senders to overload the network (which increases queue occupancy and queuing delay) before a feedback is provided to senders to reduce their rate (which takes at least as much as response time of the network) [200]. As a result, in distributed traffic control, achieving maximum throughput with minimal latency is an open problem. Using centralized schemes that perform bandwidth reservation prior to allowing senders to transmit is a candidate solution [112]. There is however several tradeoffs as discussed in Section III.

Objective Mismatch: In Section II, we presented a few objectives for traffic control. In practice, even for a single flow type, it may not be possible to optimize for all performance objectives since improving one may negatively impact the others. For example, maximizing fairness may be at odds with minimizing average latency as shown in Figure 5: the first approach is unfair but offers minimal FCT compared to second approach which is fair. Next, as shown in Figure 6, flows may have different priorities which determine their relative importance. The first approach only focuses on maximizing average throughput increasing completion time of high priority flow while the second approach minimizes FCT of high priority flow although it reduces average throughput. As another example, as stated in [86], fairness could be at odds with minimizing deadline miss rate. As a result, it is desirable to define a utility of these objectives and aim to maximize utility. In general, maximizing a utility of objective variables such as latency, utilization, deadline miss rate, lateness, and fairness is an open problem. Current research efforts mostly focus on maximizing performance with regards to one or two of these metrics. For example, the approach presented in [59] formulates an optimization scenario that considers both fairness and meeting deadlines.

VII. Related Paradigms

We review a few networking paradigms that have affected the design and operation of datacenter networks. Essentially, networks have become more flexible and controllable giving operators more room for performance optimizations.

A. Programmable Forwarding Planes

Programmable forwarding planes offer significant room for efficient control and management of datacenter networks. They make it possible to develop and implement custom policies and algorithms for network control and management. Forwarding plane can be programmed centrally using a controller that takes into account policies and resource constraints through some interface provided by forwarding elements. This is considered as part of Software Defined Networking (SDN) [201]. A comprehensive survey of SDN architecture and applications can be found in [202].

The dominant framework in this realm is OpenFlow [118], [203] where forwarding elements, such as switches, can be managed via an open interface. An important

benefit of an open interface is that switches built by different vendors can be operated in the same way allowing for cost effective expansion. Rolling out new updates to the network also becomes much easier as only controllers need to be patched. In general, there could be multiple controllers each managing a part of the network while coordinating together which is most applicable to large networks. There is two-way communication between switches and controllers: a controller can register for specific events at the switches and perform changes to the switches' forwarding table by adding new rules, modifying existing rules or removing them.

The forwarding process begins when a packet enters a switch from any port. The switch tries to match the packet to a forwarding rule according to its header fields and will forward it to the correct outgoing port. It is also possible to modify the packet contents (packet rewrite) before forwarding it. If the packet cannot be matched to any rule, it can be sent to the controller (if forwarding table is configured with the right table-miss entry) for further inspection and if necessary the controller will update the forwarding plane accordingly for forwarding of this packet and the rest of packets from the same flow. If there are multiple matches, the highest priority rule will be executed. More complex operations can be executed using Group Tables which allow for forwarding to multiple outgoing ports, selection of the outgoing port according to some hash of the packet (e.g., for load balancing), and switching to a connected output port for failover. Group Table features have been added since version 1.1 of OpenFlow [204] and currently version 1.5 has been released [205].

By centrally managing forwarding rules, one can implement different routing protocol stacks in a centralized manner. For example, BGP protocol stack can be deployed on top of SDN [206]. In addition, one can implement a more sophisticated control plane protocol that understands and communicates with a variety of other protocols, such as legacy BGP routers, while running a custom protocol stack itself [4].

B. Programmable Data Planes

Generally, data plane operations are implemented using Application-Specific Integrated Circuits (ASICs) at the hardware layer allowing for forwarding at maximum rate but only offering a set of fixed switch functions that can only be changed by replacing the ASICs. This introduces a few issues namely being prone to bugs as well as long and unpredictable time to implement new functions [207], [208]. Programmable data planes (PDPs) allow the packet processing functions to be changed at the forwarding devices, i.e., switches can apply new forwarding functions at the line rate. This is orthogonal to programmable forwarding planes (e.g., SDN) where different forwarding rules can be selectively applied to packets. PDPs make it easier to deploy traffic control schemes that depend on custom in-network processing or feedback. For example, [9], [53], [86] rely on custom packet headers and feedback from switches.

PDPs can be realized using Protocol-Independent Switch Architecture (PISA) using which new features can be introduced to switches or bug fixes can be applied in a short time [209]. There are emerging proposals for hardware designs that allow for PISA [210], [211]. Hardware prototypes (switch chips) have also been built that have made this possible [209]. P4 [212], [213] is a high level language to program PISA switches which is vendor independent, and protocol independent (i.e., operates directly on header bits and can be configured to work with any higher layer protocol). P4 compiler can also compile P4 code to run on a general purpose processor as software switches.

C. Advanced NICs

NICs have been providing basic offloading features to OSes, such as segmentation offloading, for many years. Several vendors have been developing NICs with advanced offloading features that perform complex transport tasks and deliver the results without minimal involvement from OS and CPU. These features allow complex operations at high line rates of datacenters (40 Gbps and more) doing which at the OS may incur significant CPU overhead and additional communication latency.

Examples of offloading features include cryptography, quality of service, encapsulation, congestion control, storage acceleration, erasure coding, and network policy enforcement. Examples of such NICs include Mellanox ConnectX [214] and Microsoft SmartNIC [215], [216] developed as part of Open Compute Project (OCP) [217]. SmartNIC relies on FPGA accelerators and can be used to apply SDN/Networking policies. It also makes low latency transport possible using Lightweight Transport Layer (LTL) that creates end-to-end transport connection between NICs (FPGA does all processing of segmentation, ordering, and ACKs).

D. Userspace Packet Processing

By default, packets pass through the Operating System networking stack before they are received by applications. When packets arrive at the NIC, interrupts are generated to invoke the Operating System routines that read and process them. Processing of packets is usually done in batches to reduce CPU utilization at high rates, for example by enabling Interrupt Moderation [66].

To improve the packet processing performance (decrease packet processing latency and increase throughput), a different approach would be to bypass Operating System's networking stack and use polling instead of interrupts. This can be realized using kernel bypass modules, such as Netmap [218], [219], Vector Packet Processing (VPP) [220], [221] and Data Plane Development Kit (DPDK) [222], which have been shown to reduce the number of required cycles to process a packet by up to 20$\times$ on average [218]. These modules allow userspace programs to directly access NIC buffers to read incoming packets or write packets for transmission.

Userspace networking stacks have been developed on top of kernel bypass modules. Sandstorm [223] and mTCP [224], implement TCP in userspace and rely on Netmap and VPP, respectively. SoftNIC [225] is built on top of DPDK and allows developers to program custom NIC features in software. RAMCloud [226] distributed key-value store and

FastPass [112] make use of kernel bypass and polling to speed up Remote Procedure Calls (RPC). NDP [103] is a datacenter transport protocol that is designed for ultra-low latency and operates on top of DPDK.

E. Lossless Ethernet and RDMA

TCP has been the dominant transport protocol across datacenters for the majority of applications. Since implemented as part of OS protocol stack, using TCP at high transmission rates can exhaust considerable CPU resources and impose notable amount of communication latency. Remote Direct Memory Access (RDMA) is a transport protocol that allows delivery of data from one machine to another machine without involving the OS networking protocol stack. RDMA operates on the NICs of machines communicating. Compared to TCP, RDMA offers higher bandwidth and lower latency at lower CPU utilization [8], [99], [227]. RDMA can also be used for seamless offloading of large datasets to nearby machines (as opposed to using pagefiles) [228].

To use RDMA, the underlying network has to be lossless since RDMA does not support recovery from lost data by default. Ethernet which is the favorite choice of transport in datacenters, however, does not support reliability by default. Lossless Ethernet, also known as Converged Enhanced Ethernet (CEE), supports per-hop flow control at Layer 2. Backpressure is used as the mechanism to stop senders in case of a full buffer in the network. PAUSE messages are sent to previous hops, pausing the output ports that are connected to inputs with full buffers, until the ultimate senders receive a PAUSE message. The flow control mechanism used by lossless Ethernet is referred to as Priority Flow Control (PFC) and offers 8 priority levels for various classes of traffic [154], [155].

For Layer 2 networks, RDMA can be deployed using RDMA over Converged Ethernet (RoCE) [229] which is based on lossless Ethernet and works across a single Layer 2 domain. For larger networks that span across Layer 3, RDMA can be deployed on top of IP and UDP using version 2 of RoCE (RoCEv2) [230]. It can also be deployed on top of IP and TCP using iWARP [231] which implements a full TCP stack on end-host NIC to provide a lossless end to end transport. iWARP does not require a lossless infrastructure and can work on top of usual Ethernet, but is less performant and has limited capabilities compared to RoCE [232].

Using lossless Ethernet can lead to a few performance issues in general. Some hindering issues include Layer 2 Head of Line (HOL) Blocking, unfairness (because Layer 2 has no understanding of upper layer notions such as flows), and deadlocks (due to per-port/class PAUSE feature and possible circular dependency of routes). HOL blocking might occur since pausing happens on a per-port/class basis. Therefore, a flow can overflow a port causing it to be blocked stopping other flows going through that port as well. As a result, it is necessary to prevent formation of full buffers. Furthermore, loss-based congestion control approaches are rendered useless since there is no packet loss in case of full buffers.

Quantized Congestion Notification (QCN) [233], which is fully implemented in Layer 2, can be used to reduce PAUSE messages by signaling senders before buffers are full. It sends notifications back to the sender's NIC from switches. Packets need to be tagged with a flow ID at the senders which will be used at the switches when notifications are generated to determine which flows should be slowed down. QCN is limited to boundaries of a single Layer 2 domain and therefore is insufficient for datacenters with large networks.

TCP Bolt [234] and DCQCN [8] operate across Layer 3 using RoCEv2. Both of these schemes use DCTCP [29] like ECN marking to reduce buffer occupancy and minimize PAUSE signals. To prevent deadlocks, TCP Bolt creates edge disjoint spanning trees (EDSTs) across the network with different PFC classes to prevent cyclic dependencies as flows are routed in the network. TIMELY [99] can also be used on lossless networks which uses a delay-based approach to detect increased buffer occupancy and manages its rate accordingly to reduce it.

VIII. Broader Perspective

Cloud companies and content providers, such as Google [1], Microsoft [2], Facebook [235] and Amazon [3], have built multiple datacenters in different continents and countries. Multiple datacenters offer a variety of benefits for distributed applications with geographically wide range of users such as email, multimedia (e.g., YouTube), social networks (e.g., Facebook, Instagram, and Google Plus) and online storage. These benefits include increased availability and fault-tolerance, global or regional load balancing, reduced latency to customers and reduced global bandwidth usage via caching. For example, to minimize user access latency, data can be placed on local datacenters close to users via replication.

To further improve reliability, load balancing and data availability, large datacenter operators (such as Amazon and Microsoft) operate in two hierarchies of zones and discrete datacenters. Each availability zone is usually made up of a few discrete datacenters that are close enough to communicate with negligible latency (e.g., less than 2 ms for Microsoft Azure, i.e., few tens of miles), while far enough to allow them to operate as distinct failure domains. Datacenters usually have rich connectivity within zones which themselves are connected using long haul fiber optics (usually hundreds of miles) [236]. These links are either owned by datacenter operators or leased from a provider with existing backbone infrastructure. These links that connect multiple datacenters within regions and across them are referred to as inter-datacenter networks. Maintaining and operating inter-datacenter networks requires significant capital investment from datacenter companies which makes it imperative to efficiently use them [59], [177], [237].

In general, we can categorize traffic that goes within and across datacenters into traffic that is a result of direct interaction with users and the business internal traffic that is a result of backend data processing or migration. Recent work points to significant increase in the overall business internal traffic (which includes both intra and inter-datacenter

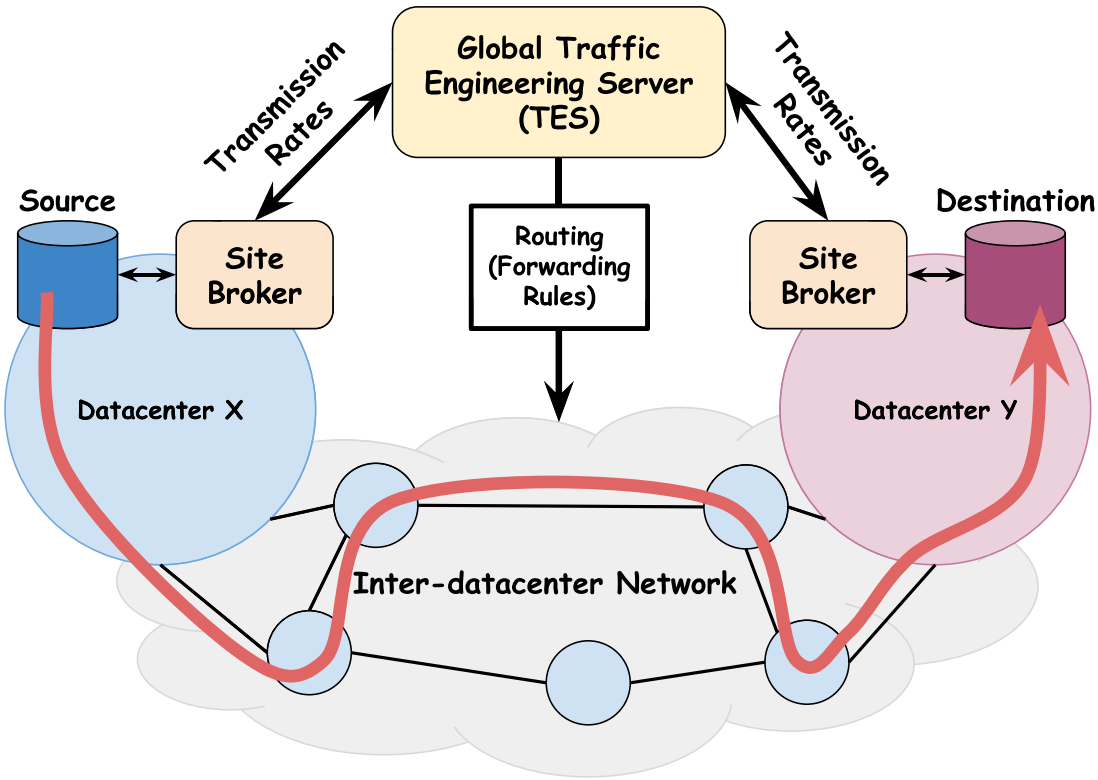


Fig. 7. Central management of private dedicated inter-datacenter networks.

traffic) that is growing at a much faster pace than user generated traffic [4], [177], [238]. Such increase not only demands higher interconnection bandwidth across servers within datacenters, but also higher network capacity across datacenters. Over the past decade, significant attention has been given to intra-datacenter networks to improve their performance and efficiency which was the topic of discussion in previous sections. However, similar attention has not been paid to increasing efficiency and performance of connectivity across datacenters.

To communicate across datacenters, many companies purchase bandwidth from ISP networks with present WAN infrastructure and are billed according to some usage criteria. A widely used pricing scheme calculates traffic costs by looking at 95 percentile of network bandwidth usage over some period of time [239]. A variety of research efforts focus on minimizing inter-datacenter traffic transit costs considering a similar pricing scheme by aiming to not increase the peak bandwidth usage or minimally increase it if necessary [240]–[248].

Large datacenter operators take advantage of dedicated inter-datacenter connections over long haul optical networks. Google operates their own backbone network referred to as B4 [20], [177]. Microsoft Global WAN [249] connects Microsoft datacenters over their private (dark) fiber network. Facebook has also developed their cross datacenter backbone network referred to as Express Backbone [238]. These private dedicated networks offer a unique opportunity for further optimization of network resource management. Considering that all end-points of such networks are managed by one organization, we can improve efficiency by coordinating transmission across such end-points. In addition, despite their huge geographical scale, these networks are usually made up of tens to hundreds of links (that connect distant locations) making such coordination practically feasible. For example, B4 is currently managed centrally by a system called Bandwidth Enforcer [250], Microsoft Global WAN is managed by SWAN [237] and Express Backbone is also managed centrally by a traffic engineering controller [238].

A. *Private Dedicated Inter-Datacenter Networks*

Bandwidth allocation is an effective approach for global traffic engineering over private dedicated networks [237], [250]. To take into account latency overhead of bandwidth allocation, a hybrid approach is usually taken where an aggregate bandwidth is set aside for short latency-sensitive flows (mostly user generated) per link or such flows are assigned strictly higher traffic priority. Large long-running flows which we refer to as transfers will then become the focus of per flow bandwidth allocation. In addition, since the network environment is continuously changing as new transfers arrive or due to varying volume of higher priority traffic (from latency-sensitive flows), a slotted timeline is considered where transmission rates can be updated on a per timeslot basis.

Figure 7 shows an example architecture for this purpose which is based on similar concepts as [237] and [250]. One way ro realize this architecture is using SDN. An inter-datacenter network that can be operated using SDN is sometimes referred to as Software Defined WAN (SDWAN).

This allows data transmission and routing to be managed centrally according to transfer properties and requirements as well as network status and topology [251].

A central Traffic Engineering Server (TES) calculates transmission rates and routes for submitted transfers as they arrive at the network. Rates are then dispatched to local agents that keep track of local transfers (initiated within the same datacenter) called site brokers. Senders first communicate with local site broker which in turn forwards transfer requests to TES. The local site broker then receives TES's response with the transmission rates and forwards it to the sender. Local site brokers add a level of indirection between senders and TES which can help in several ways. It reduces the request response overhead for TES by maintaining a persistent connection with brokers and possibly aggregating several transfer requests before relaying them to the server, which can be useful for hierarchical bandwidth allocation that is done by locally grouping many transfers and presenting them to TES as one (trading some traffic engineering accuracy for scalability). Site broker may also inform TES of network conditions within datacenters for more careful rate allocation and routing. Finally, site broker may modify TES's response according to varying local network conditions or allow senders to switch to a standby backup TES in case it goes offline. Other tasks in this system setup include the following.

Rate-limiting: We earlier discussed a variety of techniques for rate-limiting in Section V-B1. TES can calculate transmission rates on a per transfer basis in which case end-hosts (e.g., virtual machines initiating the transfers) should comply with such rates by limiting rates before traffic is transmitted on the wire. In addition, applications that initiate the transfers can themselves apply rate-limiting by carefully controlling the amount of data handed off to the transport layer. Although simple, this approach demands changes to the applications. TES can also compute rates per groups of transfers which can improve scalability [250]. Rate-limiting can then be applied at some intermediate network element via traffic shaping/policing [252]. Transfers in a group will then have to use a bandwidth sharing policy among themselves such as Fair Sharing.

Routing: Upon arrival of a transfer, forwarding routes are calculated for it by TES. Such routes are then installed in the network by adding proper forwarding rules to network's switching elements. To reduce setup overhead or save network forwarding state, TES can also reuse existing forwarding rules or aggregate several of them as one if possible. Per transfer, senders may have to attach proper forwarding labels to their packets so that they are correctly forwarded (like a Virtual LAN ID). Such labeling may also be applied transparent to the sender at an intermediate network entity (e.g., hypervisor virtual switches, backbone edge, etc).

B. Research Directions

In this section, we provide an overview of several research directions considering the scenario mentioned above.

1) Inter-Datacenter Global Rate-Allocation and Routing: The majority of inter-datacenter related research is on global optimization of inter-datacenter networks. Metrics similar to ones we discussed in Section II can be considered as objectives. For example, B4 [177] and SWAN [237] focus on maximizing utilization, Tempus [59] aims to meet transfer deadlines and maximizes minimal fraction of transfers that complete prior to deadlines in case there is not enough capacity, Amoeba [60] performs admission control for incoming traffic and only admits new transfers if their deadlines can be guaranteed and DCRoute [192] performs fast admission control to guarantee deadlines while minimizing packet reordering. The majority of prior work related to this problem either focus on delay tolerant transfers or aim at meeting deadlines while not considering completion times as a metric. For many long-running data transfer operations, completion times are important in increasing overall utility. For example, faster completion of backup operations may reduce the chance of data loss due to failures or speedy replication of data objects can improve average user's quality of experience. In addition, most prior work formulate complex optimization problems that are computationally expensive and slow especially if need be solved as transfers arrive and can increase scheduling latency. Further research is then necessary in developing global rate computation and routing algorithms that lead to solutions quickly and also consider completion times of transfers.

2) Inter-Datacenter One-to-Many Transfers: Many services run across several datacenters close to regional users to offer a better quality of experience by minimizing customer access latency (e.g., CDNs cache objects for local viewers [50], [240], [246], [253], [254]). This approach also reduces overall inter-datacenter bandwidth consumption by keeping a copy of popular objects close to users. There is also need for propagating application state to multiple locations for synchronization (e.g., search index databases [177]) or making multiple distant data copies for higher reliability and availability [251]. All of these lead to data delivery from one datacenter to multiple datacenters referred to as Point to Multipoint (P2MP) transfers [255]. P2MP data delivery over private dedicated inter-datacenter networks can be considered as a special case of multicasting for which a significant body of work is available including in-network multicasting [256], [257] and using overlay networks [258], [259]. However, there is need for coordinated schemes that improve performance by carefully selecting multicast forwarding trees and assigning transmission slots to transfers. One should also note that centralized multicasting solutions proposed for intra-datacenter networks, such as [260] and [261], may not work for inter-datacenter networks due to significant differences in topology (former is usually structured and regular while latter is not). New objectives can be considered in addition to ones proposed in Section II for P2MP transfers, such as maximizing number of receivers per transfer that complete reception in a given period of time.

3) Inter-Datacenter Failure-Aware Routing: Given their scale, inter-datacenter networks may be exposed to a variety of physical conditions and natural environments. As a result, different inter-datacenter links may have significantly different link failure probabilities (possibly by as much as three

orders of magnitude [262]). In addition, inter-datacenter traffic in general is made up of a variety of classes with different quality of service requirements and priorities [20], [50]. In general, transfers can be rerouted and rescheduled in reaction to failures. This however leads to possible service disruptions which can more seriously affect services that are sensitive to loss or latency. Given unique properties of private dedicated inter-datacenter networks, steps can be taken proactively as new transfers arrive to mitigate the effect of failures on overall utility. Several objectives can be considered per failure event including minimizing number of affected transfers upon failures considering their class of service, minimizing the amount by which latency increases across a variety of services, or maximizing utilization while leaving off spare capacity to perform quick in-network failover rerouting [263]. The latter could be applied only to a fraction of inter-datacenter traffic (i.e., more important traffic). In addition, one could study these objectives for both one-to-one and one-to-many transfer scenarios (or a mix of them).

IX. Conclusion

Many online services today depend on datacenter infrastructures to provide high availability and reliability along with scalable compute at minimal costs. Datacenter environments may be shared by many tenants and applications offering different services to their consumers (end-users). Traffic control is a necessary task in datacenter networks to efficiently use the resources and fairly share them.

In this tutorial paper, we reviewed several challenges operators, tenants and applications are faced with in datacenter traffic control. We discussed different elements of traffic control explaining the problems, challenges, proposed solutions and their tradeoffs. Despite significant research efforts, the majority of traffic control proposals are in the state of research and far from adoption in the industry considering metrics of complexity, performance and cost altogether. Further research efforts are required to reach solutions that offer higher performance at same or lower costs while maintaining low complexity.

More broadly, at the end of this paper, we also pointed to inter-datacenter communication as an evolving research area that requires further attention. We proposed a centralized architecture that captures the essence of existing solutions and discussed how efficient traffic control can be realized using it. We also pointed to three major active research areas regarding inter-datacenter communication that need further attention from the research community.

Acknowledgment

The authors would like to thank the anonymous reviewers of IEEE Communications Surveys and Tutorials for their valuable comments and suggestions that helped us significantly improve the quality of this paper. The first author would also like to especially thank David Oran, Joshua Gahm, Narayan Venkat and Atif Faheem who provided valuable intellectual support during an internship at Cisco Research Center located in Cambridge, Massachusetts, USA.

References

[1] *Google Data Center Locations*. Accessed: Dec. 27, 2017. [Online]. Available: https://google.com/about/datacenters/inside/locations

[2] *Azure Cloud Services by Location or Region*. Accessed: Dec. 27, 2017. [Online]. Available: https://azure.microsoft.com/en-us/regions/

[3] *AWS Global Infrastructure*. Accessed: Dec. 27, 2017. [Online]. Available: https://aws.amazon.com/about-aws/global-infrastructure/

[4] A. Singh *et al.*, "Jupiter rising: A decade of clos topologies and centralized control in Google's datacenter network," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 183–197, 2015.

[5] A. Greenberg *et al.*, "VL2: A scalable and flexible data center network," *Commun. ACM*, vol. 54, no. 3, pp. 95–104, Mar. 2011. [Online]. Available: http://doi.acm.org/10.1145/1897852.1897877

[6] S. Subramanian. (2015). *Network in Hyper-Scale Data Centers—Facebook*. [Online]. Available: http://www.innervoice.in/blogs/2015/03/30/network-in-hyper-scale-data-centers-facebook/

[7] G. Judd, "Attaining the promise and avoiding the pitfalls of TCP in the datacenter," in *Proc. 12th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2015, pp. 145–157.

[8] Y. Zhu *et al.*, "Congestion control for large-scale RDMA deployments," in *Proc. ACM Conf. Special Interest Group Data Commun.*, 2015, pp. 523–536. [Online]. Available: http://doi.acm.org/10.1145/2785956.2787484

[9] M. Alizadeh *et al.*, "Less is more: Trading a little bandwidth for ultra-low latency in the data center," presented at the 9th USENIX Symp. Netw. Syst. Design Implement. (NSDI), 2012, pp. 253–266.

[10] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. ACM SIGCOMM Conf. Data Commun.*, 2008, pp. 63–74. [Online]. Available: http://doi.acm.org/10.1145/1402958.1402967

[11] J. H. Ahn, N. Binkert, A. Davis, M. McLaren, and R. S. Schreiber, "HyperX: Topology, routing, and packaging of efficient large-scale networks," in *Proc. Conf. High Perform. Comput. Netw. Storage Anal.*, 2009, p. 41.

[12] C. Guo *et al.*, "Dcell: A scalable and fault-tolerant network structure for data centers," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 75–86, Oct. 2008. [Online]. Available: http://doi.acm.org/10.1145/1402946.1402968

[13] M. Alizadeh and T. Edsall, "On the data path performance of leaf-spine datacenter fabrics," in *Proc. IEEE 21st Annu. Symp. High Perform. Interconnects*, 2013, pp. 71–74.

[14] A. Valadarsky, G. Shahaf, M. Dinitz, and M. Schapira, "Xpander: Towards optimal-performance datacenters," in *Proc. 12th Int. Conf. Emerg. Netw. Exp. Technol.*, 2016, pp. 205–219.

[15] *Introducing Data Center Fabric, The Next-Generation Facebook Data Center Network*. Accessed: Dec. 27, 2017. [Online]. Available: https://code.facebook.com/posts/360346274145943/introducing-data-center-fabric-the-next-generation-facebook-data-center-network/

[16] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers randomly," presented at the 9th USENIX Symp. Netw. Syst. Design Implement. (NSDI), San Jose, CA, USA, 2012, pp. 225–238, accessed: Aug. 3, 2017. [Online]. Available: https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/singla

[17] C. Clos, "A study of non-blocking switching networks," *Bell Labs Tech. J.*, vol. 32, no. 2, pp. 406–424, Mar. 1953.

[18] J. Zhou *et al.*, "WCMP: Weighted cost multipathing for improved fairness in data centers," in *Proc. 9th Eur. Conf. Comput. Syst.*, 2014, p. 5.

[19] V. Liu, D. Halperin, A. Krishnamurthy, and T. Anderson, "F10: A fault-tolerant engineered network," in *Proc. 10th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2013, pp. 399–412.

[20] R. Govindan, I. Minei, M. Kallahalla, B. Koley, and A. Vahdat, "Evolve or die: High-availability design principles drawn from Googles network infrastructure," in *Proc. ACM SIGCOMM Conf.*, 2016, pp. 58–72.

[21] G. Porter *et al.*, "Integrating microsecond circuit switching into the data center," in *Proc. ACM SIGCOMM Conf. SIGCOMM*, 2013, pp. 447–458. [Online]. Available: http://doi.acm.org/10.1145/2486001.2486007

[22] N. Hamedazimi *et al.*, "FireFly: A reconfigurable wireless data center fabric using free-space optics," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 319–330, 2014.

[23] S. B. Venkatakrishnan, M. Alizadeh, and P. Viswanath, "Costly circuits, submodular schedules and approximate carathéodory theorems," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Model. Comput. Sci.*, 2016, pp. 75–88.

[24] M. Ghobadi *et al.*, "ProjecToR: Agile reconfigurable data center interconnect," in *Proc. Conf. ACM SIGCOMM Conf.*, 2016, pp. 216–229.

[25] A. Singla, "Fat-FREE topologies," in *Proc. 15th ACM Workshop Hot Topics Netw.*, 2016, pp. 64–70.

[26] *Cisco Global Cloud Index: Forecast and Methodology, 2015–2020 White Paper*. Accessed: Dec. 27, 2017. [Online]. Available: http://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.pdf

[27] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: Measurements & analysis," in *Proc. 9th ACM SIGCOMM Conf. Internet Meas. Conf.*, 2009, pp. 202–208.

[28] B. Vamanan, J. Hasan, and T. N. Vijaykumar, "Deadline-aware datacenter TCP (D2TCP)," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 115–126, 2012.

[29] M. Alizadeh *et al.*, "Data center TCP (DCTCP)," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 63–74, Oct. 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=2043164.1851192

[30] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. Katz, "DeTail: Reducing the flow completion time tail in datacenter networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 139–150, 2012.

[31] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[32] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: Distributed data-parallel programs from sequential building blocks," *SIGOPS Oper. Syst. Rev.*, vol. 41, no. 3, pp. 59–72, Jun. 2007. [Online]. Available: http://doi.acm.org/10.1145/1272998.1273005

[33] S. Liu, H. Xu, and Z. Cai, "Low latency datacenter networking: A short survey," *arXiv preprint arXiv:1312.3455*, 2013. [Online]. Available: https://arxiv.org/abs/1312.3455

[34] Y. Ren, Y. Zhao, P. Liu, K. Dou, and J. Li, "A survey on TCP Incast in data center networks," *Int. J. Commun. Syst.*, vol. 27, no. 8, pp. 1160–1172, 2014, doi: 10.1002/dac.2402.

[35] L. Chen, B. Li, and B. Li, "Allocating bandwidth in datacenter networks: A survey," *J. Comput. Sci. Technol.*, vol. 29, no. 5, pp. 910–917, Sep. 2014, doi: 10.1007/s11390-014-1478-x.

[36] J. Zhang, F. Ren, and C. Lin, "Survey on transport control in data center networks," *IEEE Netw.*, vol. 27, no. 4, pp. 22–26, Jul./Aug. 2013.

[37] R. K. Yadav, "TCP issues in data center system-survey," *Int. J. Adv. Found. Res. Comput.*, vol. 1, no. 1, pp. 21–25, 2014.

[38] B. Wang, Z. Qi, R. Ma, H. Guan, and A. V. Vasilakos, "A survey on data center networking for cloud computing," *Comput. Netw.*, vol. 91, pp. 528–547, Nov. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S138912861500300X

[39] R. P. Joglekar and P. Game, "Managing congestion in data center network using congestion notification algorithms," *Int. Res. J. Eng. Technol.*, vol. 3, no. 5, pp. 195–200, 2016.

[40] P. Sreekumari and J.-I. Jung, "Transport protocols for data center networks: A survey of issues, solutions and challenges," *Photon. Netw. Commun.*, vol. 31, no. 1, pp. 112–128, Feb. 2016, doi: 10.1007/s11107-015-0550-y.

[41] C. Guo *et al.*, "BCube: A high performance, server-centric network architecture for modular data centers," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 63–74, 2009.

[42] K. S. Solnushkin. *Fat-Tree Design*. Accessed: Dec. 27, 2017. [Online]. Available: http://clusterdesign.org/fat-trees/

[43] R. N. Mysore *et al.*, "PortLand: A scalable fault-tolerant layer 2 data center network fabric," in *Proc. ACM SIGCOMM Conf. Data Commun.*, 2009, pp. 39–50. [Online]. Available: http://doi.acm.org/10.1145/1592568.1592575

[44] *Cisco Data Center Spine-and-Leaf Architecture: Design Overview White Paper*. Accessed: Dec. 27, 2017. [Online]. Available: http://www.cisco.com/c/en/us/products/collateral/switches/nexus-7000-series-switches/white-paper-c11-737022.html

[45] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP extensions for multipath operation with multiple addresses," Internet Eng. Task Force, Fremont, CA, USA, RFC 6824, 2013.

[46] S. Hoory, N. Linial, and A. Wigderson, "Expander graphs and their applications," *Bull. Amer. Math. Soc.*, vol. 43, no. 4, pp. 439–561, 2006.

[47] M. Hashemi. *The Infrastructure Behind Twitter: Scale*. Accessed: Dec. 27, 2017. [Online]. Available: https://blog.twitter.com/engineering/en%5Fus/topics/infrastructure/2017/the-infrastructure-behind-twitter-scale.html

[48] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding data center traffic characteristics," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 1, pp. 92–99, 2010.

[49] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. 10th ACM SIGCOMM Conf. Internet Meas.*, 2010, pp. 267–280.

[50] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *Proc. ACM Conf. Special Interest Group Data Commun.*, 2015, pp. 123–137. [Online]. Available: http://doi.acm.org/10.1145/2785956.2787472

[51] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron, "Better never than late: Meeting deadlines in datacenter networks," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 50–61, Aug. 2011. [Online]. Available: http://doi.acm.org/10.1145/2043164.2018443

[52] S. Joy and A. Nayak, "Improving flow completion time for short flows in datacenter networks," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, May 2015, pp. 700–705.

[53] M. Alizadeh *et al.*, "pFabric: Minimal near-optimal datacenter transport," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 435–446, Oct. 2013. [Online]. Available: http://doi.acm.org/10.1145/2534169.2486031

[54] Z. Cao, M. Kodialam, and T. V. Lakshman, "Joint static and dynamic traffic scheduling in data center networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2014, pp. 2445–2453.

[55] Z. Hu, Y. Qiao, and J. Luo, "Coarse-grained traffic matrix estimation for data center networks," *Comput. Commun.*, vol. 56, pp. 25–34, Feb. 2015.

[56] L. Chen, K. Chen, W. Bai, and M. Alizadeh, "Scheduling mix-flows in commodity datacenters with karuna," in *Proc. Conf. ACM SIGCOMM*, 2016, pp. 174–187.

[57] J. Brutlag, *Speed Matters for Google Web Search*, Google, Mountain View, CA, USA, Jun. 2009.

[58] G. Linden. (2006). *Make Data Useful*. [Online]. Available: http://sites.google.com/site/glinden/Home/StanfordDataMining.2006-11-28.ppt

[59] S. Kandula, I. Menache, R. Schwartz, and S. R. Babbula, "Calendaring for wide area networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 515–526, 2015.

[60] H. Zhang *et al.*, "Guaranteeing deadlines for inter-datacenter transfers," in *Proc. 10th Eur. Conf. Comput. Syst.*, 2015, p. 20.

[61] G. DeCandia *et al.*, "Dynamo: Amazon's highly available key-value store," *ACM SIGOPS Oper. Syst. Rev.*, vol. 41, no. 6, pp. 205–220, 2007.

[62] V. Jalaparti *et al.*, "Speeding up distributed request-response workflows," in *Proc. ACM SIGCOMM Conf. SIGCOMM*, 2013, pp. 219–230. [Online]. Available: http://doi.acm.org/10.1145/2486001.2486028

[63] R. Kapoor, A. C. Snoeren, G. M. Voelker, and G. Porter, "Bullet trains: A study of NIC burst behavior at microsecond timescales," in *Proc. 9th ACM Conf. Emerg. Netw. Exp. Technol.*, 2013, pp. 133–138.

[64] L. Kleinrock, *Queueing Theory*, vol. 1. New York, NY, USA: Wiley, 1975.

[65] H. Wu, Z. Feng, C. Guo, and Y. Zhang, "ICTCP: Incast congestion control for TCP in data-center networks," *IEEE/ACM Trans. Netw.*, vol. 21, no. 2, pp. 345–358, Apr. 2013.

[66] *Interrupt Moderation*. Accessed: Dec. 27, 2017. [Online]. Available: https://msdn.microsoft.com/en-us/windows/hardware/drivers/network/interrupt-moderation

[67] Y. Geng, V. Jeyakumar, A. Kabbani, and M. Alizadeh, "Juggler: A practical reordering resilient network stack for datacenters," in *Proc. 11th Eur. Conf. Comput. Syst.*, 2016, p. 20.

[68] C. Raiciu *et al.*, "How hard can it be? Designing and implementing a deployable multipath TCP," in *Proc. 9th USENIX Conf. Netw. Syst. Design Implement.*, 2012, p. 29.

[69] W. Stevens, "TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," Internet Eng. Task Force, Fremont, CA, USA, RFC 2001, 1997.

[70] *Large Receive Offload*. Accessed: Dec. 27, 2017. [Online]. Available: https://lwn.net/Articles/243949/

[71] *Receive Side Scaling (RSS)*. Accessed: Dec. 27, 2017. [Online]. Available: https://technet.microsoft.com/en-us/library/hh997036.aspx

[72] *Generic Receive Offload*. Accessed: Dec. 27, 2017. [Online]. Available: https://lwn.net/Articles/358910/

[73] V. Jeyakumar, M. Alizadeh, D. Mazières, B. Prabhakar, and C. Kim, "EyeQ: Practical network performance isolation for the multi-tenant cloud," presented at the 4th USENIX Conf. Hot Topics Cloud Comput. (HotCloud), 2012, p. 8.

[74] A. Shieh, S. Kandula, A. Greenberg, C. Kim, and B. Saha, "Sharing the data center network," in *Proc. 8th USENIX Conf. Netw. Syst. Design Implement.*, 2011, pp. 309–322. [Online]. Available: http://dl.acm.org/citation.cfm?id=1972457.1972489

[75] L. Popa *et al.*, "ElasticSwitch: Practical work-conserving bandwidth guarantees for cloud computing," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 351–362, 2013.

[76] K. He *et al.*, "AC/DC TCP: Virtual congestion control enforcement for datacenter networks," in *Proc. ACM SIGCOMM Conf.*, 2016, pp. 244–257.

[77] B. Cronkite-Ratcliff *et al.*, "Virtualized congestion control," in *Proc. ACM SIGCOMM Conf.*, 2016, pp. 230–243.

[78] J. Hwang, J. Yoo, and N. Choi, "IA-TCP: A rate based incast-avoidance algorithm for TCP in data center networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2012, pp. 1292–1296.

[79] J. Zhang, D. Zhang, K. Huang, and Z. Qin, "Minimizing datacenter flow completion times with server-based flow scheduling," *Comput. Netw.*, vol. 94, pp. 360–374, Jan. 2016.

[80] P. X. Gao *et al.*, "pHost: Distributed near-optimal datacenter transport over commodity network fabric," in *Proc. CoNEXT*, 2015, p. 1.

[81] Y. Yang, H. Abe, K.-I. Baba, and S. Shimojo, "A scalable approach to avoid incast problem from application layer," in *Proc. IEEE 37th Annu. Comput. Softw. Appl. Conf. Workshops*, Jul. 2013, pp. 713–718.

[82] V. Vasudevan *et al.*, "Safe and effective fine-grained TCP retransmissions for datacenter communication," in *Proc. ACM SIGCOMM Conf. Data Commun.*, 2009, pp. 303–314. [Online]. Available: http://doi.acm.org/10.1145/1592568.1592604

[83] K. Zarifis *et al.*, "DIBS: Just-in-time congestion mitigation for data centers," in *Proc. 9th Eur. Conf. Comput. Syst.*, 2014, p. 6.

[84] P. Prakash, A. Dixit, Y. C. Hu, and R. Kompella, "The TCP outcast problem: Exposing unfairness in data center networks," in *Proc. 9th USENIX Conf. Netw. Syst. Design Implement.*, 2012, p. 30.

[85] N. Dukkipati and N. McKeown, "Why flow-completion time is the right metric for congestion control," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 59–62, 2006.

[86] C.-Y. Hong, M. Caesar, and P. Godfrey, "Finishing flows quickly with preemptive scheduling," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 127–138, 2012.

[87] A. Munir, I. A. Qazi, and S. B. Qaisar, "On achieving low latency in data centers," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2013, pp. 3721–3725.

[88] T. Bonald, L. Massoulié, A. Proutiére, and J. Virtamo, "A queueing analysis of max-min fairness, proportional fairness and balanced fairness," *Queueing Syst.*, vol. 53, nos. 1–2, pp. 65–84, 2006.

[89] T. Lan, D. Kao, M. Chiang, and A. Sabharwal, "An axiomatic theory of fairness in network resource allocation," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[90] D. Nace and M. Pioro, "Max-min fairness and its applications to routing and load-balancing in communication networks: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 4, pp. 5–17, 4th Quart., 2008.

[91] A. Ghodsi *et al.*, "Dominant resource fairness: Fair allocation of multiple resource types," in *Proc. 8th USENIX Conf. Netw. Syst. Design Implement.*, 2011, pp. 323–336. [Online]. Available: http://dl.acm.org/citation.cfm?id=1972457.1972490

[92] C. Joe-Wong, S. Sen, T. Lan, and M. Chiang, "Multiresource allocation: Fairness–efficiency tradeoffs in a unifying framework," *IEEE/ACM Trans. Netw.*, vol. 21, no. 6, pp. 1785–1798, Dec. 2013, doi: 10.1109/TNET.2012.2233213.

[93] B. Heller *et al.*, "ElasticTree: Saving energy in data center networks," in *Proc. 7th USENIX Conf. Netw. Syst. Design Implement.*, 2010, p. 17. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855711.1855728

[94] Y. Shang, D. Li, and M. Xu, "Energy-aware routing in data center network," in *Proc. 1st ACM SIGCOMM Workshop Green Netw.*, 2010, pp. 1–8. [Online]. Available: http://doi.acm.org/10.1145/1851290.1851292

[95] R. Wang, S. Gao, W. Yang, and Z. Jiang, "Energy aware routing with link disjoint backup paths," *Comput. Netw.*, vol. 115, pp. 42–53, Mar. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128617300270

[96] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy proportional datacenter networks," in *Proc. 37th Annu. Int. Symp. Comput. Archit.*, 2010, pp. 338–347. [Online]. Available: http://doi.acm.org/10.1145/1815961.1816004

[97] L. Wang *et al.*, "GreenDCN: A general framework for achieving energy efficiency in data center networks," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 1, pp. 4–15, Jan. 2014.

[98] K. He *et al.*, "Presto: Edge-based load balancing for fast datacenter networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 465–478, 2015.

[99] R. Mittal *et al.*, "TIMELY: RTT-based congestion control for the datacenter," in *Proc. ACM Conf. Special Interest Group Data Commun.*, 2015, pp. 537–550. [Online]. Available: http://doi.acm.org/10.1145/2785956.2787510

[100] K. Nichols and V. Jacobson, "Controlling queue delay," *Commun. ACM*, vol. 55, no. 7, pp. 42–50, 2012.

[101] R. Pan, L. Breslau, B. Prabhakar, and S. Shenker, "Approximate fairness through differential dropping," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 2, pp. 23–39, 2003.

[102] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling TCP reno performance: A simple model and its empirical validation," *IEEE/ACM Trans. Netw.*, vol. 8, no. 2, pp. 133–145, Apr. 2000.

[103] M. Handley *et al.*, "Re-architecting datacenter networks and stacks for low latency and high performance," in *Proc. Conf. ACM Special Interest Group Data Commun.*, 2017, pp. 29–42. [Online]. Available: http://doi.acm.org/10.1145/3098822.3098825

[104] I. Cho, K. Jang, and D. Han, "Credit-scheduled delay-bounded congestion control for datacenters," in *Proc. Conf. ACM Special Interest Group Data Commun.*, 2017, pp. 239–252. [Online]. Available: http://doi.acm.org/10.1145/3098822.3098840

[105] M. Alizadeh *et al.*, "CONGA: Distributed congestion-aware load balancing for datacenters," in *Proc. ACM Conf. SIGCOMM*, 2014, pp. 503–514. [Online]. Available: http://doi.acm.org/10.1145/2619239.2626316

[106] P. Wang, H. Xu, Z. Niu, D. Han, and Y. Xiong, "Expeditus: Congestion-aware load balancing in CLOS data center networks," in *Proc. 7th ACM Symp. Cloud Comput.*, 2016, pp. 442–455.

[107] D. Zhuo, Q. Zhang, V. Liu, A. Krishnamurthy, and T. Anderson, "Rack-level congestion control," in *Proc. 15th ACM Workshop Hot Topics Netw.*, 2016, pp. 148–154. [Online]. Available: http://doi.acm.org/10.1145/3005745.3005772

[108] S. Floyd, K. Ramakrishnan, and D. L. Black, "The addition of explicit congestion notification (ECN) to IP," Internet Eng. Task Force, Fremont, CA, USA, RFC 3168, 2001.

[109] W. Bai, K. Chen, L. Chen, C. Kim, and H. Wu, "Enabling ECN over generic packet scheduling," in *Proc. 12th Int. Conf. Emerg. Netw. Exp. Technol.*, 2016, pp. 191–204.

[110] P. Cheng, F. Ren, R. Shu, and C. Lin, "Catch the whole lot in an action: Rapid precise packet loss notification in data center," in *Proc. 11th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2014, pp. 17–28.

[111] B. C. Vattikonda, G. Porter, A. Vahdat, and A. C. Snoeren, "Practical TDMA for datacenter Ethernet," in *Proc. 7th ACM Eur. Conf. Comput. Syst.*, 2012, pp. 225–238.

[112] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal, "Fastpass: A centralized 'zero-queue' datacenter network," in *Proc. ACM Conf. SIGCOMM*, 2014, pp. 307–318. [Online]. Available: http://doi.acm.org/10.1145/2619239.2626309

[113] J. Perry, H. Balakrishnan, and D. Shah, "Flowtune: Flowlet control for datacenter networks," in *Proc. 14th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2017, pp. 421–435. [Online]. Available: https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/perry

[114] S. Jouet, C. Perkins, and D. Pezaros, "OTCP: SDN-managed congestion control for data center networks," in *Proc. IEEE/IFIP Netw. Oper. Manag. Symp. (NOMS)*, Apr. 2016, pp. 171–179.

[115] S. Vissicchio, O. Tilmans, L. Vanbever, and J. Rexford, "Central control over distributed routing," in *Proc. ACM Conf. Special Interest Group Data Commun.*, 2015, pp. 43–56. [Online]. Available: http://doi.acm.org/10.1145/2785956.2787497

[116] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *Proc. NSDI*, vol. 10, 2010, p. 19.

[117] A. R. Curtis, W. Kim, and P. Yalagandula, "Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 1629–1637.

[118] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[119] A. Munir *et al.*, "Minimizing flow completion times in data centers," in *Proc. IEEE INFOCOM*, 2013, pp. 2157–2165.

[120] L. Chen, S. Hu, K. Chen, H. Wu, and D. H. Tsang, "Towards minimal-delay deadline-driven data center TCP," in *Proc. 12th ACM Workshop Hot Topics Netw.*, College Park, MD, USA, 2013, p. 21.

[121] C. Ding and R. Rojas-Cessa, "DAQ: Deadline-aware queue scheme for scheduling service flows in data centers," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Sydney, NSW, Australia, 2014, pp. 2989–2994.

[122] A. Munir *et al.*, "Friends, not foes: Synthesizing existing transport strategies for data center networks," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 491–502, Aug. 2014. [Online]. Available: http://doi.acm.org/10.1145/2740070.2626305

[123] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 30, no. 4, pp. 43–56, 2000.

[124] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the performance of TCP pacing," in *Proc. IEEE INFOCOM 19th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, vol. 3. Tel Aviv, Israel, 2000, pp. 1157–1165.

[125] M. Ghobadi and Y. Ganjali, "TCP pacing in data center networks," in *Proc. IEEE 21st Annu. Symp. High Perform. Interconnects*, San Jose, CA, USA, 2013, pp. 25–32.

[126] S. Sinha, S. Kandula, and D. Katabi, "Harnessing TCP's burstiness using flowlet switching," in *Proc. ACM Hot Nets*, San Diego, CA, USA, 2004, pp. 1–6.

[127] J. Cao *et al.*, "Per-packet load-balanced, low-latency routing for Clos-based data center networks," in *Proc. 9th ACM Conf. Emerg. Netw. Exp. Technol.*, 2013, pp. 49–60.

[128] H. Xu and B. Li, "TinyFlow: Breaking elephants down into mice in data center networks," in *Proc. IEEE 20th Int. Workshop Local Metropolitan Area Netw. (LANMAN)*, Reno, NV, USA, 2014, pp. 1–6.

[129] S. Radhakrishnan, V. Jeyakumar, A. Kabbani, G. Porter, and A. Vahdat, "NicPic: Scalable and accurate end-host rate limiting," presented at the 5th USENIX Workshop Hot Topics Cloud Comput., San Jose, CA, USA, 2013, pp. 1–5. [Online]. Available: https://www.usenix.org/conference/hotcloud13/workshop-program/presentations/Radhakrishnan

[130] A. Saeed *et al.*, "Carousel: Scalable traffic shaping at end hosts," in *Proc. Conf. ACM Special Interest Group Data Commun.*, Los Angeles, CA, USA, 2017, pp. 404–417. [Online]. Available: http://doi.acm.org/10.1145/3098822.3098852

[131] K. He *et al.*, "Low latency software rate limiters for cloud networks," in *Proc. 1st Asia–Pac. Workshop Netw.*, 2017, pp. 78–84. [Online]. Available: http://doi.acm.org/10.1145/3106989.3107005

[132] H. Rodrigues, J. R. Santos, Y. Turner, P. Soares, and D. Guedes, "Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks," in *Proc. WIOV*, Portland, OR, USA, 2011, p. 6.

[133] *Cisco Application Centric Infrastructure Fundamentals*. Accessed: Dec. 27, 2017. [Online]. Available: http://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/sw/1-x/aci-fundamentals/b%5FACI-Fundamentals/b%5FACI%5FFundamentals%5FBigBook%5Fchapter%5F0100.html

[134] W. Bai *et al.*, "PIAS: Practical information-agnostic flow scheduling for data center networks," in *Proc. 13th ACM Workshop Hot Topics Netw.*, Los Angeles, CA, USA, 2014, p. 25.

[135] M. P. Grosvenor *et al.*, "Queues don't matter when you can JUMP them!" in *Proc. 12th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, Oakland, CA, USA, 2015, pp. 1–14.

[136] C. Guo *et al.*, "SecondNet: A data center network virtualization architecture with bandwidth guarantees," in *Proc. 6th Int. Conf.*, Philadelphia, PA, USA, 2010, p. 15.

[137] S. Hu *et al.*, "Providing bandwidth guarantees, work conservation and low latency simultaneously in the cloud," in *Proc. IEEE 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, San Francisco, CA, USA, Apr. 2016, pp. 1–9.

[138] *Rate Based Pacing for TCP*. Accessed: Dec. 27, 2017. [Online]. Available: http://www.isi.edu/lsam/publications/rate%5Fbased%5Fpacing/

[139] C. Raiciu *et al.*, "Improving datacenter performance and robustness with multipath TCP," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 266–277, 2011.

[140] M. Alizadeh, A. Kabbani, B. Atikoglu, and B. Prabhakar, "Stability analysis of QCN: The averaging principle," in *Proc. ACM SIGMETRICS Joint Int. Conf. Meas. Model. Comput. Syst.*, San Jose, CA, USA, 2011, pp. 49–60.

[141] C. Tian *et al.*, "Multi-tenant multi-objective bandwidth allocation in datacenters using stacked congestion control," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Atlanta, GA, USA, May 2017, pp. 1–9.

[142] Y. Cai, Y. S. Hanay, and T. Wolf, "Practical packet pacing in small-buffer networks," in *Proc. IEEE Int. Conf. Commun.*, Dresden, Germany, 2009, pp. 1–6.

[143] Y. Lu *et al.*, "One more queue is enough: Minimizing flow completion time with explicit priority notification," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Atlanta, GA, USA, May 2017, pp. 1–9.

[144] B. Pfaff *et al.*, "The design and implementation of Open vSwitch," in *Proc. 12th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, Oakland, CA, USA, 2015, pp. 117–130.

[145] N. Ek, *IEEE 802.1 P, Q-QOS on the MAC Level*, 1999. [Online]. Available: http://www.tml.tkk.fi/Opinnot/Tik-110.551/1999/papers/08IEEE802.1QosInMAC/qos.html

[146] S. Blake *et al.*, "An architecture for differentiated services," Internet Eng. Task Force, Fremont, CA, USA, RFC 2475, 1998.

[147] C. Guo *et al.*, "RDMA over commodity Ethernet at scale," in *Proc. Conf. ACM SIGCOMM Conf.*, Florianópolis, Brazil, 2016, pp. 202–215.

[148] S. Kandula, J. Padhye, and P. Bahl, "Flyways to de-congest data center networks," presented at the 8th ACM Workshop Hot Topics Netw., 2009, pp. 1–6.

[149] E. Vanini, R. Pan, M. Alizadeh, P. Taheri, and T. Edsall, "Let it flow: Resilient asymmetric load balancing with flowlet switching," in *Proc. 14th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2017, pp. 407–420. [Online]. Available: https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/vanini

[150] C. E. Hopps, "Analysis of an equal-cost multi-path algorithm," Internet Eng. Task Force, Fremont, CA, USA, RFC 2992, 2000.

[151] S. Mahapatra and X. Yuan, "Load balancing mechanisms in data center networks," in *Proc. 7th Int. Conf. Expo Emerg. Technol. Smarter World (CEWIT)*, 2010, pp. 1–6.

[152] J. Rasley *et al.*, "Planck: Millisecond-scale monitoring and control for commodity networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 407–418, 2015.

[153] *Benefits of SAN/LAN Convergence*. Accessed: Dec. 27, 2017. [Online]. Available: http://i.dell.com/sites/doccontent/business/solutions/whitepapers/en/Documents/benefits-san-lan-convergence.pdf

[154] *IEEE 802.1Qbb Priority-based Flow Control*, 2011. [Online]. Available: http://www.ieee802.org/1/pages/802.1bb.html

[155] *Priority Flow Control: Build Reliable Layer 2 Infrastructure*. Accessed: Dec. 27, 2017. [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/switches/nexus-7000-series-switches/white%5Fpaper%5Fc11-542809.pdf

[156] H. Zhang, J. Zhang, W. Bai, K. Chen, and M. Chowdhury, "Resilient datacenter load balancing in the wild," in *Proc. Conf. ACM Special Interest Group Data Commun.*, Los Angeles, CA, USA, 2017, pp. 253–266. [Online]. Available: http://doi.acm.org/10.1145/3098822.3098841

[157] A. Kabbani and M. Sharif, "Flier: Flow-level congestion-aware routing for direct-connect data centers," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Atlanta, GA, USA, May 2017, pp. 1–9.

[158] R. Zhang-Shen and N. McKeown, "Designing a fault-tolerant network using valiant load-balancing," in *Proc. IEEE 27th Conf. Comput. Commun. (INFOCOM)*, 2008, pp. 301–305.

[159] A. Dixit, P. Prakash, Y. C. Hu, and R. R. Kompella, "On the impact of packet spraying in data center networks," in *Proc. IEEE INFOCOM*, Turin, Italy, 2013, pp. 2130–2138.

[160] D. Zats *et al.*, "FastLane: Agile drop notification for datacenter networks," Dept. EECS, Univ. California at Berkeley, Berkeley, CA, USA, Rep. UCB/EECS-2013-173, 2013.

[161] S. Ghorbani, Z. Yang, P. B. Godfrey, Y. Ganjali, and A. Firoozshahian, "DRILL: Micro load balancing for low-latency data center networks," in *Proc. Conf. ACM Special Interest Group Data Commun.*, Los Angeles, CA, USA, 2017, pp. 225–238. [Online]. Available: http://doi.acm.org/10.1145/3098822.3098839

[162] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," in *Proc. 7th Symp. Oper. Syst. Design Implement.*, Seattle, WA, USA, 2006, pp. 307–320.

[163] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Proc. IEEE 26th Symp. Mass Stor. Syst. Technol. (MSST)*, 2010, pp. 1–10.

[164] V. Jalaparti *et al.*, "Network-aware scheduling for data-parallel jobs: Plan when you can," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 407–420, Aug. 2015. [Online]. Available: http://doi.acm.org/10.1145/2829988.2787488

[165] R. Grandl, G. Ananthanarayanan, S. Kandula, S. Rao, and A. Akella, "Multi-resource packing for cluster schedulers," in *Proc. ACM Conf. SIGCOMM*, Chicago, IL, USA, 2014, pp. 455–466. [Online]. Available: http://doi.acm.org/10.1145/2619239.2626334

[166] K. Jang, J. Sherry, H. Ballani, and T. Moncaster, "Silo: Predictable message completion time in the cloud," Microsoft Res., Washington, DC, USA, Rep. MSR-TR-2013-95, 2013.

[167] Q. Xie, A. Yekkehkhany, and Y. Lu, "Scheduling with multi-level data locality: Throughput and heavy-traffic optimality," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, San Francisco, CA, USA, Apr. 2016, pp. 1–9.

[168] A. Yekkehkhany, "Near data scheduling for data centers with multi levels of data locality," *arXiv preprint arXiv:1702.07802*, 2017. [Online]. Available: https://arxiv.org/abs/1702.07802

[169] A. Yekkehkhany, A. Hojjati, and M. H. Hajiesmaili, "GB-PANDAS: Throughput and heavy-traffic optimality analysis for affinity scheduling," *arXiv preprint arXiv:1709.08115*, 2017. [Online]. Available: https://arxiv.org/abs/1709.08115

[170] A. Munir, T. He, R. Raghavendra, F. Le, and A. X. Liu, "Network scheduling aware task placement in datacenters," in *Proc. 12th Int. Conf. Emerg. Netw. Exp. Technol.*, Irvine, CA, USA, 2016, pp. 221–235.

[171] P. Lahiri *et al.*, "Routing design for large scale data centers: BGP is the better IGP," presented at the NANOG, Jun. 2012, accessed: Dec. 27, 2017. [Online]. Available: https://www.nanog.org/meetings/nanog55/presentations/Monday/Lapukhov.pdf

[172] C. Filsfils *et al.*, "BGP prefix independent convergence (PIC)," Université catholique de Louvain, Louvain-la-Neuve, Belgium, Rep., p. 14, 2007. [Online]. Available: https://dial.uclouvain.be/pr/boreal/object/boreal:166710

[173] K. Agarwal, C. Dixon, E. Rozner, and J. Carter, "Shadow MACs: Scalable label-switching for commodity Ethernet," in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw.*, Chicago, IL, USA, 2014, pp. 157–162.

[174] B. Stephens, A. Cox, W. Felter, C. Dixon, and J. Carter, "PAST: Scalable Ethernet for data centers," in *Proc. 8th Int. Conf. Emerg. Netw. Exp. Technol.*, Nice, France, 2012, pp. 49–60.

[175] M. R. Nascimento, C. E. Rothenberg, M. R. Salvador, and M. F. Magalhães, "QuagFlow: Partnering quagga with OpenFlow," in *Proc. ACM SIGCOMM Conf.*, New Delhi, India, 2010, pp. 441–442. [Online]. Available: http://doi.acm.org/10.1145/1851182.1851252

[176] A. Greenberg, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Towards a next generation data center architecture: Scalability and commoditization," in *Proc. ACM Workshop Program. Routers Extensible Services Tomorrow*, Seattle, WA, USA, 2008, pp. 57–62.

[177] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined WAN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, 2013.

[178] S. A. Jyothi, M. Dong, and P. B. Godfrey, "Towards a flexible data center fabric with source routing," in *Proc. 1st ACM SIGCOMM Symp. Softw. Defined Netw. Res.*, Santa Clara, CA, USA, 2015, p. 10.

[179] N. Katta, M. Hira, C. Kim, A. Sivaraman, and J. Rexford, "HULA: Scalable load balancing using programmable data planes," in *Proc. Symp. SDN Res.*, 2016, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/2890955.2890968

[180] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in *Proc. NSDI*, vol. 11. Boston, MA, USA, 2011, pp. 99–112.

[181] M. Li, A. Lukyanenko, S. Tarkoma, and A. Yla-Jaaski, "The delayed ACK evolution in MPTCP," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Atlanta, GA, USA, 2013, pp. 2282–2288.

[182] Y. Cao, M. Xu, X. Fu, and E. Dong, "Explicit multipath congestion control for data center networks," in *Proc. 9th ACM Conf. Emerg. Netw. Exp. Technol.*, 2013, pp. 73–84.

[183] M. Kheirkhah, I. Wakeman, and G. Parisis, "MMPTCP: A multipath transport protocol for data centers," in *Proc. IEEE 35th Annu. Int. Conf. Comput. Commun. (INFOCOM)*, San Francisco, CA, USA, Apr. 2016, pp. 1–9.

[184] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

[185] V. Jacobson, K. Nichols, and K. Poduri, *RED in a Different Light*, Cisco Syst., San Jose, CA, USA, Sep. 1999.

[186] T. D. Wallace and A. Shami, "A review of multihoming issues using the stream control transmission protocol," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 2, pp. 565–578, 2nd Quart., 2012.

[187] M. Honda, Y. Nishida, L. Eggert, P. Sarolahti, and H. Tokuda, "Multipath congestion control for shared bottleneck," in *Proc. PFLDNeT Workshop*, 2009, pp. 19–24.

[188] L. Liu, D. Li, and J. Wu, "TAPS: Software defined task-level deadline-aware preemptive flow scheduling in data centers," in *Proc. 44th Int. Conf. Parallel Process. (ICPP)*, Beijing, China, 2015, pp. 659–668.

[189] H.-W. Tseng, W.-C. Chang, I.-H. Peng, and P.-S. Chen, "A cross-layer flow schedule with dynamical grouping for avoiding TCP incast problem in data center networks," in *Proc. Int. Conf. Res. Adapt. Converg. Syst.*, Odense, Denmark, 2016, pp. 91–96.

[190] S. Liu, W. Bai, H. Xu, K. Chen, and Z. Cai, "RepNet: Cutting tail latency in data center networks with flow replication," *arXiv:1407.1239*, 2014. [Online]. Available: https://arxiv.org/abs/1407.1239

[191] M. Noormohammadpour, C. S. Raghavendra, S. Rao, and A. M. Madni, "RCD: Rapid close to deadline scheduling for datacenter networks," in *Proc. World Autom. Congr. (WAC)*, Río Grande, Puerto Rico, 2016, pp. 1–6.

[192] M. Noormohammadpour, C. S. Raghavendra, and S. Rao, "DCRoute: Speeding up inter-datacenter traffic allocation while guaranteeing deadlines," in *Proc. High Perform. Comput. Data Anal. (HiPC)*, Hyderabad, India, 2016, pp. 82–90.

[193] M. Chowdhury, Y. Zhong, and I. Stoica, "Efficient coflow scheduling with varys," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 443–454, Aug. 2014. [Online]. Available: http://doi.acm.org/10.1145/2740070.2626315

[194] M. Noormohammadpour and C. S. Raghavendra, "Comparison of flow scheduling policies for mix of regular and deadline traffic in datacenter environments," *arXiv preprint arXiv:1707.02024*, 2017. [Online]. Available: https://arxiv.org/abs/1707.02024

[195] H. Leontyev and J. H. Anderson, "Tardiness bounds for FIFO scheduling on multiprocessors," in *Proc. 19th Euromicro Conf. Real Time Syst. (ECRTS)*, Pisa, Italy, Jul. 2007, p. 71.

[196] A. Wierman and B. Zwart, "Is tail-optimal scheduling possible?" *Oper. Res.*, vol. 60, no. 5, pp. 1249–1257, 2012, doi: 10.1287/opre.1120.1086.

[197] L. Schrage, "A proof of the optimality of the shortest remaining processing time discipline," *Oper. Res.*, vol. 16, no. 3, pp. 687–690, 1968.

[198] I. A. Rai, G. Urvoy-Keller, M. K. Vernon, and E. W. Biersack, "Performance analysis of LAS-based scheduling disciplines in a packet switched network," *SIGMETRICS Perform. Eval. Rev.*, vol. 32, no. 1, pp. 106–117, Jun. 2004. [Online]. Available: http://doi.acm.org/10.1145/1012888.1005702

[199] B. Kalyanasundaram and K. R. Pruhs, "Minimizing flow time nonclairvoyantly," *J. ACM*, vol. 50, no. 4, pp. 551–567, 2003.

[200] L. Jose *et al.*, "High speed networks need proactive congestion control," in *Proc. 14th ACM Workshop Hot Topics Netw.*, Philadelphia, PA, USA, 2015, pp. 1–7. [Online]. Available: http://doi.acm.org/10.1145/2834050.2834096

[201] *Software-Defined Networking (SDN) Definition*. Accessed: Dec. 27, 2017. [Online]. Available: https://www.opennetworking.org/sdn-resources/sdn-definition

[202] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1617–1634, 3rd Quart., 2014.

[203] *OpenFlow Switch Specification Version 1.0.0 (Wire Protocol 0x01)*. Accessed: Dec. 27, 2017. [Online]. Available: http://archive.openflow.org/documents/openflow-spec-v1.0.0.pdf

[204] *OpenFlow Switch Specification Version 1.1.0 Implemented (Wire Protocol 0x02)*. Accessed: Dec. 27, 2017. [Online]. Available: http://archive.openflow.org/documents/openflow-spec-v1.1.0.pdf

[205] *OpenFlow Switch Specification Version 1.5.0 (Protocol Version 0x06)*. Accessed: Dec. 27, 2017. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf

[206] *Brain-Slug: A BGP-Only SDN for Large-Scale Data-Centers*. Accessed: Dec. 27, 2017. [Online]. Available: https://www.nanog.org/sites/default/files/wed.general.brainslug.lapukhov.20.pdf

[207] *Programming the Network Data Plane*. Accessed: Dec. 27, 2017. [Online]. Available: http://netseminar.stanford.edu/seminars/03%5F31%5F16.pdf

[208] *Data Plane Programmability, the Next Step in SDN*. Accessed: Dec. 27, 2017. [Online]. Available: http://sites.ieee.org/netsoft/files/2017/07/Netsoft2017%5FKeynote%5FBianchi.pdf

[209] *Barefoot Worlds Fastest and Most Programmable Networks*. Accessed: Dec. 27, 2017. [Online]. Available: https://barefootnetworks.com/resources/worlds-fastest-most-programmable-networks/

[210] P. Bosshart *et al.*, "Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN," in *Proc. ACM SIGCOMM Conf. SIGCOMM*, Hong Kong, 2013, pp. 99–110. [Online]. Available: http://doi.acm.org/10.1145/2486001.2486011

[211] A. Sivaraman *et al.*, "Packet transactions: High-level programming for line-rate switches," in *Proc. ACM SIGCOMM Conf.*, Florianópolis, Brazil, 2016, pp. 15–28. [Online]. Available: http://doi.acm.org/10.1145/2934872.2934900

[212] P. Bosshart *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, 2014.

[213] *P4*. Accessed: Dec. 27, 2017. [Online]. Available: http://p4.org/

[214] *Ethernet Cards—Overview*. Accessed: Dec. 27, 2017. [Online]. Available: http://www.mellanox.com/page/ethernet%5Fcards%5Foverview

[215] D. Firestone, "SmartNIC: FPGA innovation in OCS servers for microsoft azure," *OCP US Summit*, 2016, accessed: Dec. 27, 2017. [Online]. Available: https://ocpussummit2016.sched.com/event/68u4/smartnic-fpga-innovation-in-ocs-servers-for-microsoft-azure

[216] A. Greenberg, "SDN for the cloud," in *Proc. Keynote ACM Conf. Special Interest Group Data Commun.*, 2015, accessed: Dec. 27, 2017. [Online]. Available: https://conferences.sigcomm.org/sigcomm/2015/pdf/papers/keynote.pdf

[217] *Open Compute Project*. Accessed: Dec. 27, 2017. [Online]. Available: http://opencompute.org/

[218] L. Rizzo and M. Landi, "Netmap: Memory mapped access to network devices," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 422–423, 2011.

[219] *Netmap—A Framework for Fast Packet I/O*. Accessed: Dec. 27, 2017. [Online]. Available: https://github.com/luigirizzo/netmap

[220] *FD.io*. Accessed: Dec. 27, 2017. [Online]. Available: https://fd.io/technology

[221] *Vector Packet Processing*. Accessed: Dec. 27, 2017. [Online]. Available: https://github.com/chrisy/vpp

[222] *DPDK*. Accessed: Dec. 27, 2017. [Online]. Available: http://dpdk.org/

[223] I. Marinos, R. N. M. Watson, and M. Handley, "Network stack specialization for performance," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 175–186, 2014.

[224] E. Y. Jeong *et al.*, "mTCP: A highly scalable user-level TCP stack for multicore systems," in *Proc. 11th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, Seattle, WA, USA, 2014, pp. 489–502.

[225] S. Han *et al.*, "SoftNIC: A software NIC to augment hardware," EECS Dept., Univ. California at Berkeley, Berkeley, CA, USA, Rep. UCB/EECS-2015-155, May 2015. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-155.html

[226] J. Ousterhout *et al.*, "The RAMCloud storage system," *ACM Trans. Comput. Syst.*, vol. 33, no. 3, p. 7, 2015.

[227] A. Dragojević, D. Narayanan, O. Hodson, and M. Castro, "FaRM: Fast remote memory," in *Proc. 11th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, Seattle, WA, USA, 2014, pp. 401–414.

[228] J. Gu, Y. Lee, Y. Zhang, M. Chowdhury, and K. G. Shin, "Efficient memory disaggregation with infiniswap," in *Proc. 14th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2017, pp. 649–667. [Online]. Available: https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/gu

[229] *Supplement to Infiniband Architecture Specification Volume 1, Release 1.2.1 Annex A16: RDMA Over Converged Ethernet (RoCE)*, Infiniband Trade Assoc., Beaverton, OR, USA, Apr. 2010. [Online]. Available: https://cw.infinibandta.org/document/dl/7148

[230] *Supplement to InfiniBand Architecture Specification Volume 1, Release 1.2.2 Annex A17: RoCEv2 (IP Routable RoCE)*, Infiniband Trade Assoc., Beaverton, OR, USA, Sep. 2014. [Online]. Available: https://cw.infinibandta.org/document/dl/7781

[231] R. Sharp, H. Shah, F. Marti, A. Eiriksson, and W. Noureddine, "Remote direct memory access (RDMA) protocol extensions," Internet Eng. Task Force, Fremont, CA, USA, RFC 7306, 2014.

[232] *RoCE vs. iWARP Competitive Analysis*. Accessed: Dec. 27, 2017. [Online]. Available: http://www.mellanox.com/related-docs/whitepapers/WP%5FRoCE%5Fvs%5FiWARP.pdf

[233] R. Pan, B. Prabhakar, and A. Laxmikantha, *QCN: Quantized Congestion Notification: An Overview*, 2007, accessed: Dec 27, 2017. [Online]. Available: http://www.ieee802.org/1/files/public/docs2007/au_prabhakar_qcn_overview_geneva.pdf

[234] B. Stephens *et al.*, "Practical DCB for improved data center networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Toronto, ON, Canada, 2014, pp. 1824–1832.

[235] *Facebook in Los Lunas: Our Newest Data Center*. Accessed: Dec. 27, 2017. [Online]. Available: https://code.facebook.com/posts/337730426564112/facebook-in-los-lunas-our-newest-data-center/

[236] M. Russinovich. (2017). *Inside Microsoft Azure Datacenter Hardware and Software Architecture*. [Online]. Available: https://myignite.microsoft.com/videos/54962

[237] C.-Y. Hong *et al.*, "Achieving high utilization with software-driven WAN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 15–26, 2013.

[238] *Building Express Backbone: Facebook's New Long-Haul Network*. Accessed: Dec. 27, 2017. [Online]. Available: https://code.facebook.com/posts/1782709872057497/building-express-backbone-facebook-s-new-long-haul-network/

[239] *95th Percentile Bandwidth Metering Explained and Analyzed*. Accessed: Dec. 27, 2017. [Online]. Available: http://www.semaphore.com/blog/94-95th-percentile-bandwidth-metering-explained-and-analyzed

[240] Y. Feng and B. Li, "Jetway: Minimizing costs on inter-datacenter video traffic," in *Proc. ACM Int. Conf. Multimedia*, Nara, Japan, 2012, pp. 259–268.

[241] Y. Wang, S. Su, A. X. Liu, and Z. Zhang, "Multiple bulk data transfers scheduling among datacenters," *Comput. Netw.*, vol. 68, pp. 123–137, Aug. 2014.

[242] S. Su, Y. Wang, S. Jiang, K. Shuang, and P. Xu, "Efficient algorithms for scheduling multiple bulk data transfers in inter-datacenter networks," *Int. J. Commun. Syst.*, vol. 27, no. 12, pp. 4144–4165, 2014.

[243] T. Nandagopal and K. P. N. Puttaswamy, "Lowering inter-datacenter bandwidth costs via bulk data scheduling," in *Proc. Cluster Cloud Grid Comput. (CCGrid)*, Ottawa, ON, Canada, 2012, pp. 244–251.

[244] N. Laoutaris, G. Smaragdakis, R. Stanojevic, P. Rodriguez, and R. Sundaram, "Delay-tolerant bulk data transfers on the Internet," *IEEE/ACM Trans. Netw.*, vol. 21, no. 6, pp. 1852–1865, Dec. 2013.

[245] Y. Feng, B. Li, and B. Li, "Postcard: Minimizing costs on inter-datacenter traffic with store-and-forward," in *Proc. Int. Conf. Distrib. Comput. Syst. Workshops*, Macau, China, 2012, pp. 43–50.

[246] Y. Lin, H. Shen, and L. Chen, "EcoFlow: An economical and deadline-driven inter-datacenter video flow scheduling system," in *Proc. 23rd ACM Int. Conf. Multimedia*, Brisbane, QLD, Australia, 2015, pp. 1059–1062.

[247] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez, "Inter-datacenter bulk transfers with netstitcher," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 74–85, 2011.

[248] V. Jalaparti, I. Bliznets, S. Kandula, B. Lucier, and I. Menache, "Dynamic pricing and traffic engineering for timely inter-datacenter transfers," in *Proc. Conf. ACM SIGCOMM Conf.*, Florianópolis, Brazil, pp. 73–86, 2016. [Online]. Available: http://doi.acm.org/10.1145/2934872.2934893

[249] *How Microsoft Builds Its Fast and Reliable Global Network*. Accessed: Dec. 27, 2017. [Online]. Available: https://azure.microsoft.com/en-us/blog/how-microsoft-builds-its-fast-and-reliable-global-network/

[250] A. Kumar *et al.*, "BwE: Flexible, hierarchical bandwidth allocation for WAN distributed computing," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 1–14, 2015.

[251] X. Zhao, V. Vusirikala, B. Koley, V. Kamalov, and T. Hofmeister, "The prospect of inter-data-center optical networks," *IEEE Commun. Mag.*, vol. 51, no. 9, pp. 32–38, Sep. 2013.

[252] *Comparing Traffic Policing and Traffic Shaping for Bandwidth Limiting*. Accessed: Dec. 27, 2017. [Online]. Available: https://www.cisco.com/c/en/us/support/docs/quality-of-service-qos/qos-policing/19645-policevsshape.html

[253] V. K. Adhikari, S. Jain, Y. Chen, and Z.-L. Zhang, "Vivisecting YouTube: An active measurement study," in *Proc. INFOCOM*, Orlando, FL, USA, 2012, pp. 2521–2525.

[254] R. Meshenberg, N. Gopalani, and L. Kosewski. (2013). *Active-Active for Multi-Regional Resiliency*. [Online]. Available: http://techblog.netflix.com/2013/12/active-active-for-multi-regional.html

[255] M. Noormohammadpour, C. S. Raghavendra, S. Rao, and S. Kandula, "DCCast: Efficient point to multipoint transfers across datacenters," in *Proc. 9th USENIX Workshop Hot Topics Cloud Comput. (HotCloud)*, 2017, pp. 1–8. [Online]. Available: https://www.usenix.org/conference/hotcloud17/program/presentation/noormohammadpour

[256] M. Cotton, L. Vegoda, and D. Meyer, "IANA guidelines for IPv4 multicast address assignments," Internet Eng. Task Force, Fremont, CA, USA, RFC 5771, 2010.

[257] B. Quinn and K. Almeroth, "IP multicast applications: Challenges and solutions," Internet Eng. Task Force, Fremont, CA, USA, RFC 3170, 2001.
[258] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proc. SIGCOMM*, Pittsburgh, PA, USA, 2002, pp. 205–217.
[259] Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 8, pp. 1456–1471, Oct. 2002.
[260] A. Iyer, P. Kumar, and V. Mann, "Avalanche: Data center multicast using software defined networking," in *Proc. COMSNETS*, Bengaluru, India, 2014, pp. 1–8.
[261] J. Cao *et al.*, "Datacast: A scalable and efficient reliable group data delivery service for data centers," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 12, pp. 2632–2645, Dec. 2013.
[262] M. Ghobadi and R. Mahajan, "Optical layer failures in a large backbone," in *Proc. Internet Meas. Conf.*, Santa Monica, CA, USA, 2016, pp. 461–467. [Online]. Available: http://doi.acm.org/10.1145/2987443.2987483
[263] H. H. Liu, S. Kandula, R. Mahajan, M. Zhang, and D. Gelernter, "Traffic engineering with forward fault correction," in *Proc. SIGCOMM*, Chicago, IL, USA, 2014, pp. 527–538.

Mohammad Noormohammadpour (S'17) received the bachelor's degree in electrical engineering from the Sharif University of Technology, in 2014 and the master's degree in computer science from the University of Southern California (USC) in 2017, where he is currently pursuing the Ph.D. degree with the Ming Hsieh Department of Electrical Engineering. His research is focused on improving the performance and reliability of data transfers across geographically dispersed datacenters. He was a recipient of the Fellowship Award from USC in 2014. In 2016, he interned with the Cisco Innovation Center, Cambridge, MA, USA, where he worked on distributed coded storage systems. His research interests are networked systems, datacenters, and software defined networking.

Cauligi S. Raghavendra (F'97) received the B.Sc. degree in physics from Bangalore University in 1973, the B.E. and M.E. degrees in electronics and communication from the Indian Institute of Science, Bangalore, in 1976 and 1978, respectively, and the Ph.D. degree in computer science from the University of California at Los Angeles in 1982. He is a Professor of electrical engineering and computer science and the Vice Dean of the Global Academic Initiatives, Viterbi School of Engineering, University of Southern California (USC), Los Angeles. He was a Faculty Member with the Department of Electrical Engineering-Systems, USC from 1982 to 1992, as a Boeing Chair Professor of computer engineering with the School of Electrical Engineering and Computer Science, Washington State University, Pullman, from 1992 to 1997, and at The Aerospace Corporation from 1997 to 2001. He was the Chairman of EE-Systems Department from 2003 to 2005, the Senior Associate Dean of Academic Affairs from 2005 to 2006, and Strategic Initiatives from 2006 to 2011. His research interests include wireless and sensor networks, parallel and distributed systems, and machine learning. He was a recipient of the Presidential Young Investigator Award in 1985.