Congestion Control for Cross-Datacenter Networks

Gaoxiong Zeng, Wei Bai, Ge Chen, Kai Chen, *Senior Member, IEEE*, Dongsu Han, *Member, IEEE*, Yibo Zhu, and Lei Cui

***Abstract*— Geographically distributed applications hosted on cloud are becoming prevalent. They run on *cross-datacenter network* that consists of multiple data center networks (DCNs) connected by a wide area network (WAN). Such a cross-DC network poses significant challenges in transport design because the DCN and WAN segments have vastly distinct characteristics (*e.g.*, buffer depths, RTTs). In this paper, we find that existing DCN or WAN transport reacting to ECN or delay alone do not (and cannot be extended to) work well for such an environment. The key reason is that neither of the signals, by itself only, can simultaneously capture the location and degree of congestion, mainly due to the discrepancies between DCN and WAN. Motivated by this, we present the design and implementation of GEMINI that strategically integrates both ECN and delay signals for cross-DC congestion control. To achieve low latency, GEMINI bounds the inter-DC latency with delay signal and prevents the intra-DC packet loss with ECN. To maintain high throughput, GEMINI modulates the window dynamics and maintains low buffer occupancy utilizing both congestion signals. GEMINI is implemented in Linux kernel and evaluated by extensive testbed experiments. Results show that GEMINI achieves up to 53%, 31%, 76% and 2% reduction of small flow average completion times, and up to 34%, 39%, 9% and 58% reduction of large flow average completion times compared to TCP Cubic, DCTCP, BBR and TCP Vegas.**

***Index Terms*— DCN, WAN, transport, congestion control.**

I. INTRODUCTION

APPLICATIONS running in geographically distributed setting are becoming prevalent [2]–[9]. Large-scale online services often share or replicate their data into multiple DCs in different geographic regions. For example, a retailer website runs a database of in-stock items replicated in each regional data center for fast serving local customers. These regional databases synchronize with each other periodically for the latest data. Other examples include image sharing on online social networks, video storage and streaming, geo-distributed data analytics, etc.

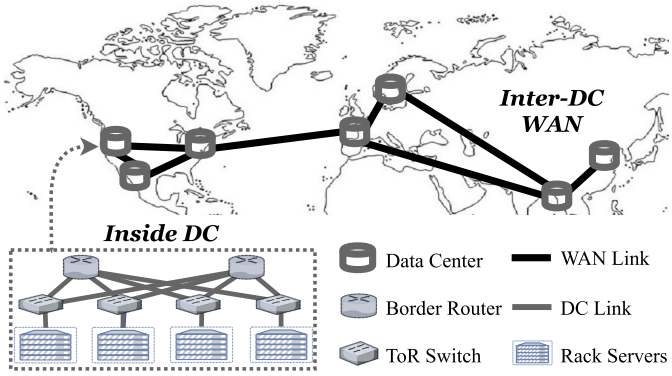


Fig. 1. Cross-datacenter network.

These applications run on *cross-datacenter (DC) network* (Figure 1) that consists of multiple data center networks (DCNs) connected by a wide area network (WAN). The wide area and intra-DC networks have vastly distinct characteristics (§II-A). For WAN, achieving high network utilization is a focus and switches have deep buffers. In contrast, latency is critical in DCN and switches have shallow buffers. While there are numerous transport protocols designed for either DCN or WAN individually, to the best of our knowledge, very little work has considered a cross-DC environment consisting of both parts at the same time.

To handle congestion control in either DCN or WAN, existing solutions have leveraged either ECN (*e.g.*, DCTCP [10] and DCQCN [11]) or delay (*e.g.*, Vegas [12] and TIMELY [13]) as the congestion signal, and successfully delivered compelling performance in terms of high-throughput and low-latency [10]–[16]. Unfortunately, due to the discrepancies between DCN and WAN, none of existing solutions designed for DCN or WAN works well for a cross-DC network (§II-B). Even worse, it is unlikely, if not impossible, that they can be easily extended to work well.

The fundamental reason is that these solutions only exploit one of the signals (either ECN or delay), which suffices

Manuscript received October 5, 2019; revised February 5, 2021; accepted March 7, 2022; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor N. B. Shroff. This work was supported in part by the Hong Kong RGC TRS under Grant T41-603/20-R, Grant GRF-16215119, and Grant GRF-16213621. This work is an extended version of an ICNP'19 paper [1] [DOI: 10.1109/ICNP.2019.8888042]. *(Corresponding author: Kai Chen.)*

Gaoxiong Zeng was with the iSING Laboratory, The Hong Kong University of Science and Technology, Hong Kong. He is now with Huawei Technologies, Shenzhen 518129, China (e-mail: gzengaa@connect.ust.hk).

Wei Bai was with the iSING Laboratory, The Hong Kong University of Science and Technology, Hong Kong. He is now with Microsoft Research Laboratory, Redmond, WA 98052 USA (e-mail: baiwei0427@gmail.com).

Ge Chen was with the iSING Laboratory, The Hong Kong University of Science and Technology, Hong Kong. He is now with ByteDance, Shanghai 201100, China (e-mail: gchenaj@connect.ust.hk).

Kai Chen is with the iSING Laboratory, The Hong Kong University of Science and Technology, Hong Kong (e-mail: kaichen@cse.ust.hk).

Dongsu Han is with the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 305-701, South Korea (e-mail: dongsuh@ee.kaist.ac.kr).

Yibo Zhu is with ByteDance, Shanghai 201100, China (e-mail: bobzhuyb@gmail.com).

Lei Cui is with Huawei Technologies, Shenzhen 518129, China (e-mail: cuilei107@huawei.com).

Digital Object Identifier 10.1109/TNET.2022.3161580

TABLE I
BUFFER SIZE FOR COMMODITY DCN SWITCHES AND WAN ROUTERS

	DCN			WAN	
Switch / Router	Arista 7010T	Arista 7050T	Arista 7050QX	Arista 7504R	Arista 7516R
Capacity (ports×BW)	48×1Gbps	48×10Gbps	32×40Gbps	576×10 Gbps/144×100Gbps	2304×10Gbps/576×100Gbps
Total buffer size	4 MB	9 MB	12 MB	96 GB	384 GB
Buffer per port per Gbps	85 KB	19.2 KB	9.6 KB	16.7 / 6.7 MB	16.7 / 6.7 MB

for a relatively homogeneous environment. However, by their nature, ECN or delay alone cannot handle heterogeneity. First, ECN is difficult to configure to meet requirements of mixed flows. The inter-DC and intra-DC flows coexist in cross-DC network, with RTTs varying by up to 1000×. Small RTT flows require lower ECN thresholds for low latency; while large RTT flows require larger ones for high throughput. In fact, tuning ECN threshold may not work, because DC switch shallow buffers can be easily overwhelmed by bursty large-BDP cross-DC traffic. For example, DCN can account for $4-20\times$ more packet losses than WAN in experiments under realistic workload (see Table II). Moreover, ECN may not be well supported in WAN.

Meanwhile, delay signal, by itself, is limiting in *simultaneously* detecting congestion in WAN and DCN. Cross-DC flows may congest either in WAN or DCN, while delay signal cannot distinguish them given its end-to-end nature. This leads to a dilemma of either under-utilizing WAN (deep-buffered) links with small delay thresholds or increasing DCN (shallow-buffered) packet losses with higher thresholds. For example, Vegas, when scaling its default parameters by 20, achieves $1.5\times$ higher throughput at the cost of $>30\times$ more intra-DC packet losses. Furthermore, low delay thresholds impose harsh requirements on accurate delay measurement [13], for which extra hardware support is needed.

The above problems call for a new synergy that considers not just one of, but both ECN and delay signals in congestion control for cross-DC network communications. Specifically, the new solution must be able to handle the following key challenges (§III-A) that have not been exposed to any of prior works: (1) How to achieve persistent low latency in the heterogeneous environment, even if DC switches (more likely to drop packet) and WAN routers (more likely to accumulate large buffering) have vastly different buffer depths. (2) How to maintain high throughput for inter-DC traffic with shallow-buffered DC switches, even if the propagation delay is in tens of milliseconds range, instead of $<250\,\mu$s assumed by DCN transport protocols such as DCTCP.

Toward this end, this paper presents GEMINI to organically integrate ECN and delay through the following three main ideas (§III-B) to combat the above two challenges:

- *Integrating ECN and Delay Signals for Congestion Detection:* Delay signal is leveraged to bound the total in-flight traffic over the entire network path including the WAN segment, while ECN signal is used to control the per-hop queue inside DCN. With bounded end-to-end latency and limited packet losses, persistent low latency is guaranteed.
- *Modulating the ECN-Triggered Window Reduction Aggressiveness by the RTT of a Flow:* Unlike conventional TCPs that drain queues more for larger RTT flows, we make large RTT flows decrease rates more gently, resulting in smoother "sawtooth" window dynamics. This, in turn, prevents bandwidth under-utilization of inter-DC traffic, while sustaining low ECN threshold for intra-DC traffic.
- *Adapting to RTT Variation in Window Increase:* We scale the additive window increase step in proportion to RTT, which better balances the convergence speed and system stability under mixed inter-DC and intra-DC traffic.

Finally, we evaluate GEMINI by extensive testbed experiments (§IV). We implement GEMINI with Linux kernel 4.9.25 and commodity switches. We show that GEMINI achieves up to 49% higher throughput compared to DCTCP under DCN congestion, and up to 87% lower RTT compared to Cubic under WAN congestion; converges to bandwidth fair-sharing point in a quick and stable manner regardless of different RTTs; and delivers persistent low flow completion times (FCT)—up to 53%, 31%, 76% and 2% reduction of small flow average completion times, and up to 34%, 39%, 9% and 58% reduction of large flow average completion times compared to TCP Cubic, DCTCP, BBR and TCP Vegas.

II. BACKGROUND AND MOTIVATION

We show heterogeneity of cross-DC networks in §II-A, and demonstrate transport performance impairments in §II-B.

A. Heterogeneity in Cross-DCNs

The real-world cross-datacenter networks present heterogeneous characteristics in the following aspects:

Heterogeneous Networking Devices: A cross-DC network consists of heterogeneous networking devices (*e.g.*, with distinct buffer depths) from intra-DC network (DCN) and inter-DC WAN. Table I gives a survey of switches or routers [17] commonly used in DCN and WAN. DCN switches have shallow buffers, up to tens of kilobytes per port per Gbps. In contrast, WAN routers adopt deep buffers, up to tens of megabytes per port per Gbps.

Mixed Intra-DC and Inter-DC Traffic: Intra-DC and inter-DC traffic coexists in the cross-DC network [18], [19]. They exhibit very different RTTs. To demonstrate this, we conduct RTT measurements on one of the major cloud platforms with 12 representative DCs across the globe. Figure 2 shows the result. The intra-DC RTTs are as small as hundreds of microseconds. In contrast, the inter-DC RTTs vary from several milliseconds to hundreds of milliseconds.

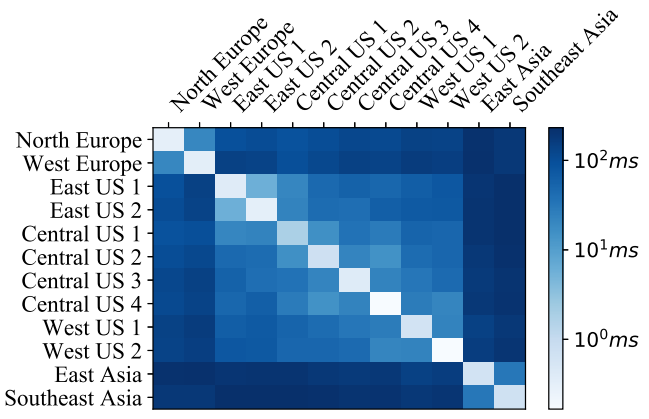


Fig. 2. RTT heat map in Cross-DC network.

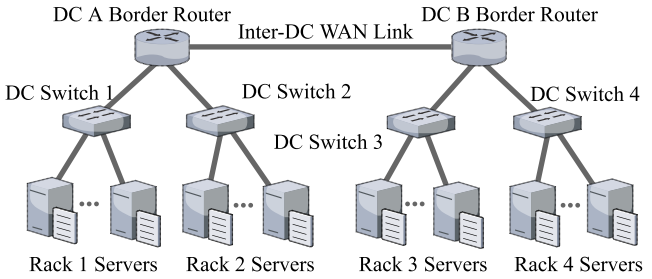


Fig. 3. Cross-datacenter network testbed.

Different Administrative Control: Cloud operators have full control over DCN, but do not always control the WAN devices. This is because many cloud operators lease the network resource (*e.g.*, guaranteed bandwidth) from Internet service providers (ISPs) and WAN gears are maintained by the ISPs. As a result, some switch features, *e.g.*, ECN, may not be well supported [20], [21] (either disabled or configured with undesirable marking thresholds) in WAN.

The heterogeneity imposes great challenges in transport design. Ideally, transport protocols should take congestion location (buffer depth), traffic type (RTT) and supported mechanism (*e.g.*, ECN) into consideration. We show how prior designs are impacted without considering the heterogeneity in the following subsection (§II-B).

B. Single Signal's Limitations With Heterogeneity

Most of the existing transport protocols [10]–[15] use either ECN or delay as the congestion signal. While they may work well in either DCN or WAN, we find that ECN or delay alone cannot handle heterogeneity. We conduct extensive experiments to study the performance impairments of leveraging ECN or delay signal alone in cross-DC networks.

Testbed: We build a testbed (Figure 3) that emulates 2 DCs connected by an inter-DC WAN link. Each DC has 1 border router, 2 DC switches and 24 servers. All links have 1 Gbps capacity. The intra-DC and inter-DC base RTTs (without queueing) are $\sim 200\,\mu$s and ~ 10 ms,[1] respectively. The maximum per-port buffer size of DC switch and border router are ~450 and 10,000 1.5 KB-MTU-sized packets, respectively.

Schemes Experimented Instead of enumerating every transport protocol, we select several transport solutions that are representative for their own category based on the congestion signal and are readily deployable with solid Linux kernel implementation. Specifically, we experiment Cubic [23], Vegas [12], BBR [14] and DCTCP [10]. Cubic is experimented with and without ECN. ECN threshold at DC switches is set to 300 packets[2] to guarantee high throughput for inter-DC traffic (as suggested by Figure 5(b)). ECN is not enabled in the WAN segment. Vegas uses two parameters α and β to control the lower and upper bound of excessive packets in flight. We experiment the default setting ($\alpha = 2, \beta = 4$) and scaled by 10 settings ($\alpha = 20, \beta = 40$).

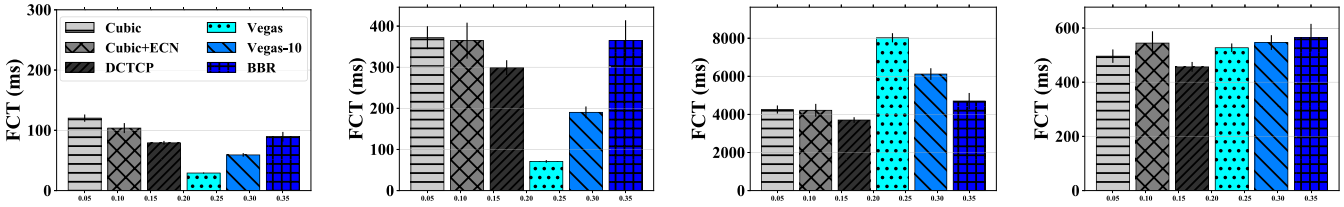
We run realistic workload based on a production trace of web search [10]. All flows cross the inter-DC WAN link. The average utilization of the inter-DC and intra-DC links are ~90% and ~11.25–45%. The flow completion time (FCT) results are shown in Figure 4. We make the following observations and claims, and elaborate them later in the section:

- Transport protocols based on loss or ECN signal only (*e.g.*, Cubic, Cubic + ECN and DCTCP) perform poorly in small flow FCTs (Figure 4(a) and 4(b)). This is because they experience high packet losses in shallow-buffered DCN (Table II) and large queueing delay without ECN in WAN. We further find that configuring ECN threshold is fundamentally difficult under *mixed traffic*.
- Transport protocols based on delay signal, when using small thresholds (*e.g.*, Vegas), achieve good performance for small flows (Figure 4(a) and 4(b)) at the cost of slowing down large flows (Figure 4(c)). In contrast, when using large thresholds (*e.g.*, Vegas with the scaled by 10 parameters), they greatly degrade the performance of small flows. We further demonstrate the dilemma on setting delay thresholds under *distinct buffer depths*.
- BBR suffers from high packet loss rates (> 0.1%), leading to poor small flow FCTs (Figure 4(a) and 4(b)). BBR requires precise estimates of available bandwidth and RTT, which is difficult to achieve under dynamic workload. For example, the bandwidth probing based on a multiple-phase cycle (8 RTTs by default) may not catch up with bursty traffic quickly enough to avoid buffer overflow. And it does not explicitly react to loss signal, leading to continuous losses until the mismatched estimation expires.

Problems of ECN-Signal-Only Solutions: ECN-based transport uses the ECN signal [24], [25] that often reflects the exceeding queue length at the congested network link. For it to deliver high throughput, switch should not mark ECN until

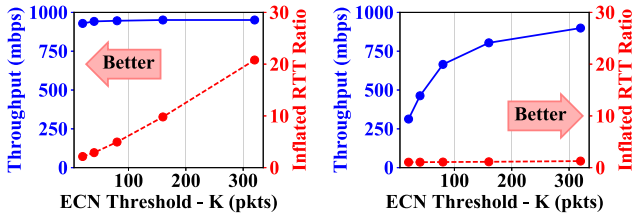
[1]Our DC border routers are emulated by servers with multiple NICs, so that we can use NETEM [22] to emulate inter-DC propagation delay.

[2]We have tuned the ECN threshold for both DCTCP and Cubic. The selected ECN marking threshold achieves the best throughput. A higher one leads to higher loss rate and thus lower throughput. A lower one also results in lower throughput due to frequent congestion notification.



(a) Small Flow - Average (b) Small Flow - 99th Tail (c) Large Flow - Average (d) All Flow - Average

Fig. 4. Flow completion time (FCT) results. Small flow: Size $<$ 100 KB. Large flow: Size $>$ 10 MB.



(a) Intra-DC Flows: *RTT*=200 μs (b) Inter-DC Flows: *RTT*=10 ms

Fig. 5. Conflicting ECN requirements in DCTCP. The right y-axis shows latency by the inflated RTT ratio—the queueing-inflated RTT normalized by the base RTT (w/o queueing).

queue length reaches the bandwidth-delay product (BDP) of the network path[3] or a constant fraction of it [10], [26], [27]. *However, in a cross-DC setting, it is difficult to configure the marking parameters due to the large difference in RTT among different paths and divergent requirements imposed by intra-DC and inter-DC flows.* Intra-DC flows impose small buffer pressure but have stringent latency requirement (*e.g.*, hundreds of microseconds). In contrast, inter-DC flows have looser latency requirement given the large base latency of WAN, instead require large buffer space for high WAN utilization.

To demonstrate the problem, we generate incast flows from hosts in the same rack to a remote server using DCTCP. We perform two experiments in this setting. In the first experiment, we choose a destination server in the same DC so there are intra-DC flows only. In the second experiment, we choose a destination server in a remote DC so there are inter-DC flows only. In both cases, the bottleneck link is at the source DC switch due to the incast traffic pattern. We vary the ECN marking threshold of the bottleneck switch between 20, 40, 80, 160, and 320 packets per port.

Figure 5(a) and 5(b) show the throughput and latency results of intra-DC and inter-DC flows, respectively. From Figure 5(a), we observe a small threshold is desirable to achieve low latency for intra-DC flows. In contrast, from Figure 5(b), we observe inter-DC flows require a high threshold for high throughput. Clearly, there is a conflict: one cannot achieve high throughput and low latency simultaneously for both inter-DC and intra-DC flows in the cross-DC network.

In fact, achieving high utilization over cross-DC is non-trivial because intra-DC switches have shallow buffers — the shallow buffer is easily overwhelmed by bursty large-BDP cross-DC flows (we call it *buffer mismatch*). We confirm that by measuring the packet loss rate (PLR) in previous dynamic workload experiments. Table II shows the results. We find that packet losses happen within DCN mostly ($>$ 80%), even though inter-DC WAN is more heavily loaded than intra-DC links. The high losses then lead to low throughput for loss-sensitive protocols. Large-BDP cross-DC traffic is a key factor of the problem. We repeat the same experiments with the inter-DC link delay set to 0. All traffic is now with low BDPs. We observe small PLRs ($< 10\times10^{-5}$) within DCN for all ECN-based schemes this time. Further, we find that naively pacing packets like in BBR cannot completely resolve the problem. For example, Cubic with FQ/pacing [28] has similar high PLR (66×10^{-5}) in DCN compared to raw Cubic.

TABLE II
DCN/WAN PACKET LOSS RATE (10^{-5})

Cubic	Cubic + ECN	DCTCP
78 / 10	24 / 6	19 / $<$ 1

In addition, ECN-based transport protocols require ECN marking support from all network switches. However, ECN marking may not be well supported. It is either disabled or configured with undesirable marking thresholds in WAN (discussed in §II-A). As a result, ECN-based transport such as DCTCP may fall back on using packet loss signal, leading to high packet losses and long queueing delay.

Problems of delay-signal-only solutions: Delay-based transports use the delay signal [12], [13] that reflects the cumulative end-to-end network delay. Typically, they have a threshold to control the total amount of in-flight traffic. *However, given different buffer depths in WAN and DCN, a dilemma arises when setting the delay threshold — either inter-DC throughput or intra-DC latency is sacrificed.*

Cross-DC flows may face congestion either in WAN or DCN. Delay signal handles both indistinguishablly given its end-to-end nature. On the one hand, if we assume congestion occurs in WAN, the delay thresholds should be large enough (usually in proportion to the BDP) to fully utilize the WAN bandwidth. However, if the bottleneck resides in the DCN instead, the large thresholds (*e.g.*, 10 ms $\times$ 1 Gbps = 1.25 MB) can easily exceed the DC switch shallow buffers (*e.g.*, 83 KB per Gbps) and cause frequent packet losses. On the other hand, if we assume congestion happens in DCN, the delay thresholds should be low enough (at least bounded by the DC switch buffer sizes) to avoid severe intra-DC packet losses. However, if the bottleneck resides in WAN instead, the low thresholds

[3]BDP is an attribute of a network path calculated by multiplying the bottleneck link bandwidth and the zero-queueing round-trip delay. Thus, BDP varies for flows of different network paths.

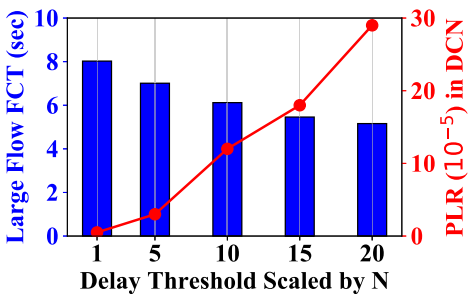


Fig. 6. Dilemma in setting delay threshold. The left y-axis shows throughput by the flow completion time (FCT) of large flows. The right y-axis shows packet loss rate (PLR) inside DCN.

can greatly impair the bandwidth utilization. In sum, the dilemma of setting delay thresholds arises.

To demonstrate the problem, we run the same benchmark workloads used earlier in the section. We experiment Vegas with the default setting ($\alpha = 2, \beta = 4$) and scaled by N settings ($\alpha = 2 \times N, \beta = 4 \times N$), where N is set to 1, 5, 10, 15, 20. Results are shown in Figure 6. On the one hand, small delay thresholds degrade the inter-DC throughput, leading to high average FCT for large flows. On the other hand, large delay thresholds increase packet losses significantly in shallow-buffered DCN. Therefore, setting the delay thresholds are faced with a dilemma of either hurting inter-DC throughput or degrading intra-DC packet loss rate.

In addition, low delay thresholds impose harsh requirement over accurate delay measurement, for which extra device supports (*e.g.*, NIC prompt ACK in [13]) are needed.

III. GEMINI

We introduce our design rationale in §III-A, describe the detailed GEMINI congestion control algorithm in §III-B, and provide guidelines for setting parameters in §III-C.

A. Design Rationale

How to Achieve Persistent Low Latency in the Heterogeneous Network Environment?: Persistent low latency implies low end-to-end queueing delay and near zero packet loss. Obviously, ECN, as a per-hop signal, is not a good choice for bounding the end-to-end latency; not to mention, ECN has limited availability in WAN. If we use delay signal alone, small delay threshold is necessary for low loss given the DC switch shallow buffer. However, with a small amount of in-flight traffic, we may not be able to fill the network pipe of the WAN segment (demonstrated in §II-B).

Instead of using a single type of signal alone, we integrate ECN and delay signals to address this challenge. In particular, delay signal, given its end-to-end nature, is effectively used to bound the total in-flight traffic; and ECN signal, as a per-hop signal, is leveraged to control the per-hop queues. Aggressive ECN marking is performed at the DC switch to prevent shallow buffer overflow. Thus, the constraint of using small delay thresholds is removed, leaving more space to improve WAN utilization. In this way, the aforementioned dilemma of delay-based transport is naturally resolved.

How to Maintain High Throughput for Inter-DC Traffic With Shallow-Buffered DC Switches?: A majority of transport (*e.g.*, DCTCP) follow additive-increase multiplicative-decrease (AIMD) congestion control rule. The queue length they drain in each window reduction is proportionate to BDP ($C \times RTT$) [10], [26], [27]. Essentially, the queue length drained each time should be smaller than the switch buffer size to avoid buffer empty and maintain full throughput. Thus, given large RTT range in cross-DC network, high buffers are required. In deep-buffered WAN, setting a moderately high delay threshold works well to balance throughput and latency. However, in shallow-buffered DCN, aggressive ECN marking is required for low queueing and low loss rate. With limited buffer space, sustaining high throughput gets extremely difficult (demonstrated in §II-B).

To address this buffer mismatch challenge, we modulate the aggressiveness of ECN-triggered window reduction by RTT. Maintaining high throughput, in effect, requires large RTT flows to drain queues as small as small RTT flows do during window reduction. Intuitively, we make larger RTT flows reduce rates more gently, thus resulting in smoother "sawtooth" window and queue length dynamics. In this way, bandwidth under-utilization can be effectively mitigated, while still using a small ECN marking threshold. The use of small ECN threshold leave enough headroom in the shallow buffer switches because it keeps the average buffer occupancy low, reducing the delay and packet drop.

Further, we adjust the window increase step in proportion to BDP. Conventional AIMD adopts fixed constant window increase step for all flows. This either hurts convergence speed of large-BDP inter-DC flows, or makes the system unstable for small-BDP intra-DC flows. When BDP is large, AIMD requires more RTTs to climb to the peak rate, leading to slower convergence. In contrast, when BDP is small, AIMD may frequently overshoot the bottleneck bandwidth, resulting in more frequent losses and thus less stable performance. Therefore, we adjust the window increase step in proportion to BDP for better robustness under heterogeneity.

B. GEMINI Algorithm

GEMINI is a window-based congestion control algorithm that uses additive-increase and multiplicative-decrease (AIMD). Following the design rationale above, GEMINI leverages both ECN and delay signals for congestion detection. It further adjusts the extent of window reduction as well as growth function based on RTTs of the flows to incorporate heterogeneity. The GEMINI algorithm is summarized by flowchart in Figure 7 and pseudocode in Algorithm 1. Parameters and variables are summarized in Table III.

Integrating ECN and Delay for Congestion Detection: The congestion detection mechanism leverages both ECN and delay signals. Delay signal is used to bound the total in-flight traffic in the network pipe. ECN signal is used to control the per-hop queues inside DCN. By integrating ECN and delay signal, low latency can be achieved [29]. Specifically, DCN congestion is detected by ECN, so as to meet the stringent per-hop queueing control requirement imposed by shallow

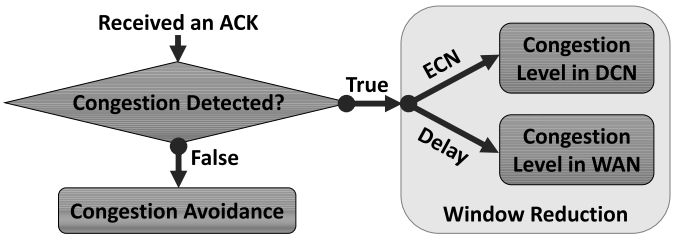


Fig. 7. GEMINI congestion control process.

Algorithm 1 GEMINI Congestion Control Algorithm

Input : New Incoming ACK
Output: New Congestion Window Size

```
/* Update transport states (e.g., α)   */
update_transport_state(α, rtt_base, rtt_min) ;
/* When congested, set 1; else 0.      */
congested_dcn  ←  ecn_indicated_congestion() ;
congested_wan  ←  rtt_indicated_congestion() ;
if congested_dcn || congested_wan then
    if time since last cwnd reduction > 1 RTT then
        F     ←  4 × k/(c × rtt_base + k) ;
        f_dcn ←  α × F × congested_dcn ;
        f_wan ←  β × congested_wan ;
        cwnd  ←  cwnd × (1 - max(f_dcn, f_wan)) ;
else
    h     ←  H × c × rtt_base ;
    cwnd  ←  cwnd + h/cwnd ;
```

TABLE III
PARAMETERS AND VARIABLES USED IN GEMINI

Parameter	Description
K	ECN marking threshold
T	Delay threshold
β	Parameter for window reduction in WAN
H	Parameter for congestion window increase
Variable	**Description**
$CWND$	Congestion window
f_dcn	Extent of window reduction in DCN
f_wan	Extent of window reduction in WAN
RTT_{min}	Minimum RTT observed in previous RTT
RTT_{base}	Minimum RTT observed during a long time
RTT	Simplified notation of RTT_{base}
C	Bandwidth capacity (constant for given network)
α	Average fraction of ECN marked packets
F	Scale factor for DCN congestion control
h	Adaptive congestion window increase step

buffers. WAN congestion is detected by delay, because the end-to-end delay is dominated mostly in WAN than in DCN.[4]

[4]The delay signal cannot exclude the DCN queueing delay. However, DCN queueing is often low due to DCN shallow buffer, much lower than that on WAN. Such a low DCN queueing delay has very limited impact with a relative large delay threshold as shown in Table V.

DCN congestion is indicated by the ECN signal — the ECN-Echo flag set in the ACKs received by the senders. The ECN signal is generated exactly the same as DCTCP. Data packets are marked with Congestion Experienced (CE) code-point when instantaneous queueing exceeds marking threshold at the DC switches. Receivers then echo back the ECN marks to senders through ACKs with the ECN-Echo flags. Given shallow-buffered DCN, the ECN signal is leveraged with a small marking threshold for low packet losses.

WAN congestion is indicated by the delay signal — ACKs returned after data sending with persistent larger delays: $RTT_{min} > RTT_{base} + T$, where RTT_{min} is the minimum RTT observed in previous RTT (window); RTT_{base}, or simplified as RTT, is the base RTT (minimum RTT observed during a long time); T is the delay threshold. Inspired by [30], we use RTT_{min} instead of average or maximum RTTs, which can better detect persistent queueing and tolerate transient queueing possibly caused by bursty traffic. Given deep-buffered WAN, the delay signal is used with a moderately high threshold for high throughput and bounded end-to-end latency.

When either of the two signals indicate congestion, we react to the signal by reducing the congestion window correspondingly. When both ECN and delay signals indicate congestion, we react to the one of heavier congestion:

$$CWND = CWND \times (1 - max(f\_dcn, f\_wan))$$

where f_dcn determines the extent of window reduction for congestion in DCN; and f_wan determines that of WAN. We show how to compute them later in the section.

Modulating the ECN-Triggered Window Reduction Aggressiveness by RTT: The window reduction algorithm aims to maintain full bandwidth utilization while reducing the network queueing as much as possible. This essentially requires switch buffer never underflow at the bottleneck link. Given distinct buffer depths, GEMINI reduces congestion window differently for congestion in DCN and WAN.

In DCN, given shallow buffer, strictly low ECN threshold is used for low packet losses. We adopt the DCTCP algorithm, which works well under the low ECN threshold for the intra-DC flows. However, for large RTT inter-DC flows, the throughput drops greatly. This is because the buffer drained by a flow during window reduction increases with its RTT (*e.g.*, the amplitude of queue size oscillations for DCTCP is $O(\sqrt{C \times RTT})$ [10], [27]). Larger RTT flows drain queues more and easily empty the switch buffers, leading to low link utilization. Inspired by this, GEMINI extends DCTCP by modulating the window reduction aggressiveness based on RTT. This guides the design of f_dcn — the extent of window reduction when congestion is detected in DCN. When ECN signal indicates congestion, we compute f_dcn as follows:

$$f\_dcn = \alpha \times F$$

where α is the exponential weighted moving average (EWMA) fraction of ECN marked packets, F is the factor that modulates the congestion reduction aggressiveness. We derive the scale factor $F = \frac{4K}{C \times RTT + K}$ (see Theorem 1), where C is the bandwidth capacity (a constant parameter for given network), RTT is the minimum RTT observed during a long time, K is the

ECN marking threshold. Thus, for intra-DC flows, following the guideline in DCTCP by setting $K = (C \times \textit{RTT})/7$, we have $F = \frac{1}{2}$, exactly matching the DCTCP algorithm. For inter-DC flows with larger RTTs, F gets smaller, leading to smaller window reduction and smoother queue length oscillation.

In WAN, given much deeper buffer, high throughput can be more easily maintained than in DCN. In fact, window reduction based on a fixed constant, like standard TCPs [23], [31] do, is enough for high throughput. There are potentially a wide range of threshold settings to effectively work with (see §III-C). This guides the design of f_wan — the extent of window reduction when congestion is detected in WAN. When RTT signal indicates congestion, we compute f_wan as follows:

$$f\_wan = \beta$$

where β is a window decrease parameter for WAN.

The window reduction is performed no more than once per RTT, which is the minimum time required to get feedback from the network under the new sending rate. Despite the congestion detection by ECN and delay, packet losses and timeouts may still occur. For that, we keep the same fast recovery and fast retransmission mechanism from TCP.

Window Increase That Adapts to RTT Variation: The congestion avoidance algorithm adapts to RTTs (or BDP when the bandwidth capacity is fixed) to help balance convergence speed and stability. For conventional AIMD, large BDP flows need more RTTs to climb to the peak rate, leading to slow convergence; while small BDP flows may frequently overshoot the bottleneck bandwidth, leading to unstable performance. Therefore, adjusting the window increasing step in proportion to BDP compensates the RTT variation, and makes the system more robust under diverse RTTs. Further, it also mitigates RTT unfairness [32], [33], which in turn helps to improve tail performance. This leads to the adaptive congestion window increase factor h. When there is no congestion indication, for each ACK,

$$CWND = CWND + \frac{h}{CWND}$$

h is a congestion avoidance factor in proportion to BDP: $h = H \times C \times \textit{RTT}$, where H is a constant parameter, C is the bandwidth capacity, *RTT* is the minimum RTT observed during a long time. We prove that factor h together with the scale factor F guarantees bandwidth fair-sharing regardless of different RTTs in Appendix §B.

Summary: GEMINI resolves the conflicting requirements imposed by network heterogeneity naturally by integrating ECN and delay signal, specifically, (1) in face of *distinct buffer depths*, GEMINI handles congestion in WAN and DCN by delay and ECN signal respectively, simultaneously meeting the need of strictly low latency in DCN and high bandwidth utilization in WAN; (2) in face of *mixed traffic* with large range of RTTs in shallow-buffered DCN, GEMINI maintains high throughput by modulating the window reduction aggressiveness based on RTTs. In particular, large RTT flows reduce the windows more gently, effectively avoiding buffer empty and bandwidth under-utilization. This is achieved by scale factor F, which guarantees full throughput under limited buffer space or small ECN threshold at steady state (see Theorem 1 with detailed proof in Appendix §A). Besides, Gemini adapts its window increase step in proportion to the RTT, achieving faster convergence speed and better fairness.

TABLE IV
DEFAULT GEMINI PARAMETER SETTINGS

Parameter	Default Value
K	$50\ pkts\ /\ Gbps$
T	$5\ ms$
β	0.2
H	1.2×10^{-7}

Theorem 1: Given a positive ECN marking threshold K, we can maintain 100% throughput under DCN congestion if congestion window is reduced as follows,

$$CWND = CWND \times (1 - \alpha \times F)$$

where α is the EWMA of ECN fraction and $F \leq \frac{4K}{C \times RTT + K}$.

C. Guidelines for Setting Parameters

Default GEMINI parameter settings are shown in Table IV. We adopt the default parameter settings throughout all our experiments unless otherwise specified. We provide the following rules of thumbs for setting the parameters, but leave finding the optimal threshold settings to the future work.

ECN Marking Threshold (K): The scaling factor F ensures full link utilization given an ECN threshold (K). As a lower K indicates a smaller queue, setting K as low as possible may seem desirable. However, there is actually a trade-off here. When K is small, the scaling factor F is also small, making the flows reduce their congestion window slowly, leading to slower convergence. Therefore, we recommend a moderately small threshold of 50 packets per Gbps. In addition, to mitigate the effect of packet bursts (especially for large BDP inter-DC traffic), we use a per-flow rate limiter at the sender to evenly pace out each packet.

Queueing Delay Threshold (T): T should be sufficiently large to achieve high throughput in the cross-DC pipe. It should also leave enough room to filter out the interference from the DCN queueing delay. In practice (§II-A), RTTs (include queueing) in production DCNs are at most 1ms. We recommend to set $T = 5\,ms$ that is higher enough to remove the potential DCN queueing interference (see Table V).

Window Decrease Parameter (β): GEMINI reduces the window size by β multiplicatively when WAN congestion is detected. To avoid bandwidth under-utilization, we need to have queueing headroom $T > \frac{\beta}{1-\beta}\textit{RTT}$, or $\beta < \frac{T}{T+\textit{RTT}}$ based on the buffer sizing theory [26]. Thus, we have $\beta < 0.33$, assuming $\textit{RTT} = 10ms$ and $T = 5ms$. We recommend to set $\beta = 0.2$ (the same reduction factor as Cubic and Vegas) for smoother 'sawtooth'. In practice, this is stricter than necessary as competing flows are often desynchronized [26]. We show that the recommended T and β settings can well serve the cross-DC networks in a wide range of RTTs in §IV-B.

Window Increase Parameter (H): In congestion avoidance phase, GEMINI grows its congestion window size by h MSS

every RTT. In our implementation, we actually scale h with BDP ($C \times RTT$) instead of RTT only, that is, $h = H \times C \times RTT$. This is reasonable as large BDP means potentially large window size. Scaling h with BDP achieves better balance between convergence speed and stability. We recommend to set $H = 1.2 \times 10^{-7}$ with bounded minimum/maximum increase speed of 0.1 / 5 respectively as a protection. This leads to $h = 1$ when $C = 1Gbps$ and $RTT = 8ms$, a middle ground between large BDP inter-DC traffic and low BDP intra-DC traffic.

IV. Evaluation

In this section, we present the detailed GEMINI Linux kernel implementation and evaluation setup in §IV-A, and conduct extensive experiments to answer the following questions:

§IV-B Does GEMINI *Achieve High Throughput and Low Latency?:* We show that GEMINI achieves higher throughput ($1-1.5\times$) and equally low delay compared to DCTCP under DCN congestion; lower delay ($>7\times$) and equally high throughput compared to Cubic under WAN congestion.

§IV-C Does GEMINI *Converge Quickly, Fairly and Stably?:* In static traffic experiments, we show that GEMINI converges to the bandwidth fair-sharing point quickly and stably under both DCN congestion and WAN congestion, regardless of distinct RTTs differed by up to 64 times.

§IV-D How Does GEMINI *Perform Under Realistic Workload?:* In realistic traffic experiments, we show that under both cases (intra-DC heavy or inter-DC heavy traffic pattern), GEMINI persistently achieves the one of the best flow completion times for both short and large flows.

A. Implementation and Experiment Setup

GEMINI *Implementation:* GEMINI is developed based on Linux kernel 4.9.25. Linux TCP stack has a universal congestion control interface defined in struct $tcp\_congestion\_ops$, which supports various pluggable congestion control modules. The congestion window reduction algorithm is implemented in $in\_ack\_event()$ and $ssthresh()$. The congestion avoidance algorithm is implemented in $cong\_avoid()$.

Testbed: Experiments are conducted in 2 testbeds with 1Gbps and 10Gbps capacity respectively. The 1Gbps testbed has a larger scale than the 10Gbps one. Both testbeds share the same topology as shown in Figure 3. There are 2 data centers connected by an inter-DC WAN link. Each data center has one border router, two DC switches and multiple servers. Border routers are emulated by servers with multiple NICs, so that we can use NETEM [22] to emulate WAN propagation delay. Intra-DC (under single ToR) and inter-DC base RTTs are $\sim 200\,\mu$s and ~ 10 ms, respectively. Dynamic buffer allocation [34] at the DC switches is enabled like most operators do in real deployments to absorb bursts.

- *Large-scale 1Gbps Testbed:* There are 50 Dell PowerEdge R320 servers and 4 Pica8 P-3297 switches. Pica8 P-3297 switches have 4MB buffer shared by 48 ports. The WAN buffer is set to 10,000 1.5 KB-MTU-sized packets per port. All network interfaces are set to 1Gbps full duplex mode.
- *Small-scale 10Gbps Testbed:* There are 10 HUAWEI RH1288 V2 servers and 1 Mellanox SN2100 switch (divided into multiple VLANs). Mellanox SN2100 switches have 16MB buffer shared by 16 ports. The WAN buffer is set to 80,000 1.5 KB-MTU-sized packets per port. All network interfaces are set to 10Gbps full duplex mode.

Remark: We show results of the large-scale testbed by default.

Schemes Compared: We experiment Cubic [23], Vegas [12], BBR [14], DCTCP [10] and GEMINI. All these protocols have implementations in Linux kernel TCP and are readily deployable in practice. *Cubic* is the default loss-based congestion control algorithm used in Linux system. It is experimented with and without ECN. *DCTCP* is an ECN-based congestion control algorithm designed to achieve high throughput, low latency and high burst tolerance in DCN. The ECN marking threshold is set to 300 packets to guarantee high throughput for inter-DC traffic. *Vegas* uses two parameters α and β to control the lower and upper bound of excessive packets in flight. We experiment the default setting ($\alpha = 2, \beta = 4$) and scaled by 10 settings ($\alpha = 20, \beta = 40$) to show the throughput and latency trade-off. *BBR* is designed primarily for the enterprise WAN. It tries to drive the congestion control to the theoretical optimal point [35] with maximized throughput and minimized latency, based on accurate bandwidth and RTT estimation. GEMINI is the transport design proposed in this paper. Default GEMINI parameter settings are shown in Table IV. We adopt the default parameter settings throughout all experiments in this paper if not specified. The ECN marking is configured only at the DC switches.

B. Throughput and Latency

We show that GEMINI achieves high throughput and low latency under both DCN congestion and WAN congestion.

Handling Congestion in DCN: ECN-based DCN congestion control module needs to cope with the mismatch between DC switch shallow buffer and high-BDP inter-DC traffic so as to strike a good balance between latency and throughput. We show that, by adding BDP-aware scale factor F, the mismatch issue can be mitigated to a great extent.

To demonstrate that, we generate many-to-one long flows sharing one DC switch bottleneck link. We perform two experiments, with all intra-DC flows in the first one and all inter-DC flows in the second. The RTT-based WAN congestion control module is disabled here for GEMINI (the module will not work even if we enable it, because the RTT threshold itself will filter out the DCN congestion). We set the ECN marking threshold K to 20, 40, 80, 160, 320 packets.

Results show that there is little gap between GEMINI and DCTCP for the intra-DC flows. The average RTTs of inter-DC flows are also similar (so the results are neglected here). The throughputs of inter-DC flows are shown in Figure 8. GEMINI maintains slightly higher throughput (938 mbps) than DCTCP (899 mbps) when setting K as high as 320 packets. Setting

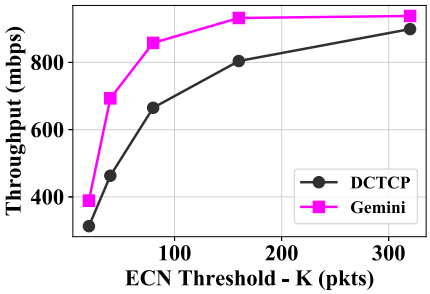


Fig. 8. Aggregate throughput of inter-DC flows that bottlenecked at a DCN link. GEMINI is less buffer-hungry (requires 0–76% smaller K) than DCTCP when achieving similar throughput.

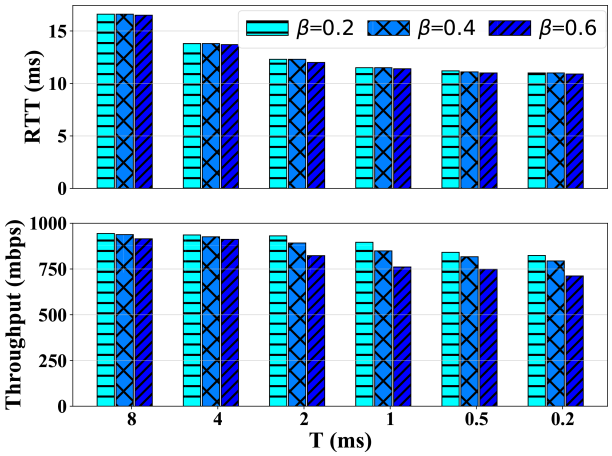


Fig. 9. RTT and throughput of inter-DC flows bottlenecked at a WAN link.

TABLE V
IMPACT OF RTT NOISE ON THROUGHPUT

Avg. RTT Noise (ms)	0	0.1	0.5	1
Throughput (Mbps)	944	944	936	927

a higher threshold is prohibitive given limited buffer left to avoid packet losses under bursty traffic.

Handling Congestion in WAN: GEMINI leverages delay signal for WAN congestion control. To quantify the signal error, we measure RTTs in our testbed under a quiescent scenario. The standard deviation of the measured intra-rack and inter-DC RTT are 17 μs and 58 μs, respectively. To show how noisy RTT can impact GEMINI, we add RTT estimation errors deliberately in our GEMINI kernel module and run many-to-one static flows sharing one bottleneck link in WAN. The random noise is added to each RTT sample with uniform distribution in the range of [0, x] ms, where x is set to 0, 0.2, 1, 2. The results are listed in Table V. The noise from the variable kernel processing time ($<$ 0.1 ms) has no impact to the GEMINI throughput. When considering the potential DCN queueing delay interference ($<$ 1 ms), the aggregate throughput of GEMINI only drops slightly. This verifies that setting $T = 5\,ms$ can well filter out the interference from DCN queueing delay. We also attribute the robustness partially to the design that uses RTT_{min} in each window time to detect persistent congestion.

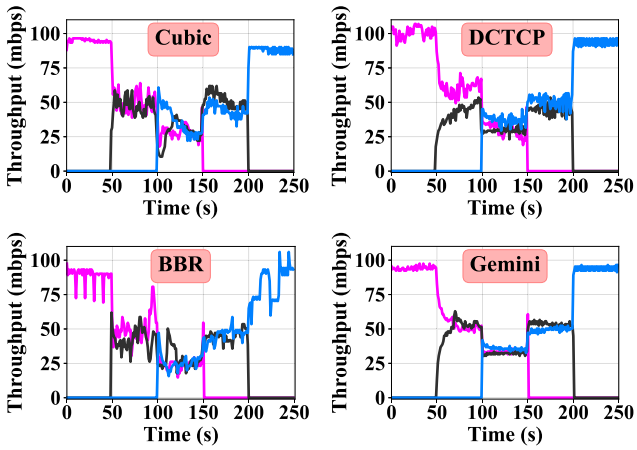


Fig. 10. GEMINI converges quickly, fairly and stably.

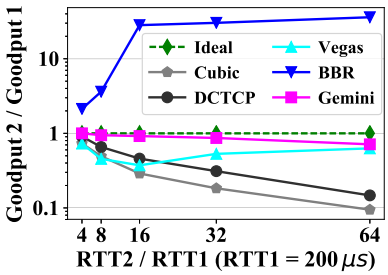


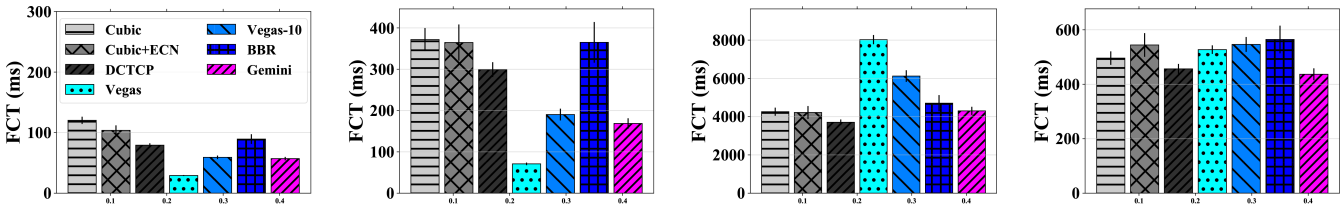
Fig. 11. RTT-fairness. RTT1 and RTT2 are the intra-DC and inter-DC RTTs. GEMINI achieves bandwidth fair-sharing regardless of different RTTs.

To further demonstrate the effectiveness of the congestion control in WAN, we run many-to-one static flows sharing one bottleneck link in WAN, with varying T and β settings. The results are shown in Figure 9. In general, GEMINI maintains near full throughput with average queueing-delayed RTTs of 16 ms (highest when $T = 8ms$ and $\beta = 0.2$). Compared to the transport protocols that leverage loss signals in WAN, GEMINI achieves similar high throughput at the cost of much lower latency. For example, in another experiment with same setup, Cubic suffers from 7$\times$ higher average RTTs ($\sim$ 100 ms).

Parameter Sensitivity: The GEMINI performance mainly relies on two parameters, *i.e.*, ECN marking threshold K and queueing delay threshold T for congestion control in DCN and WAN, respectively. We now analyze the performance of GEMINI under various parameter settings.

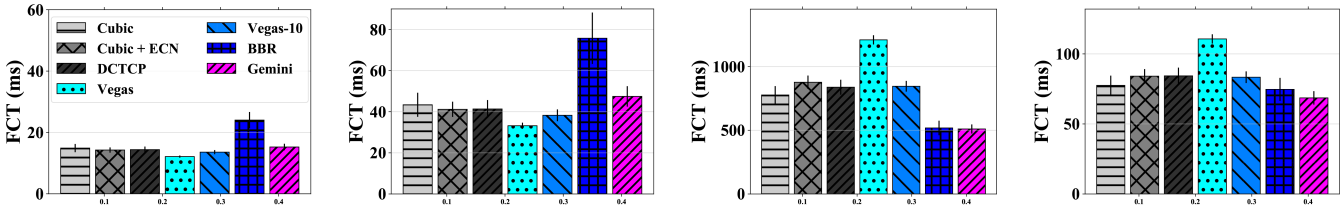
Figure 8 shows the GEMINI performance with varying K under DCN congestion. We can see that GEMINI performs better than DCTCP in a large range of parameter settings. In fact, the GEMINI throughput is not degraded until the threshold K is set to as low as 100 packets. This means that GEMINI is less buffer-hungry (requires 0-76% smaller K) than DCTCP when achieving similar throughput, leaving enough space to improve burst tolerance and latency of intra-DC traffic.

Figure 9 shows the GEMINI performance with varying T and β under WAN congestion. As expected, a lower T leads to lower RTT latency at the cost of slightly decreased



(a) Small Flow - Average (b) Small Flow - 99th Tail (c) Large Flow - Average (d) All Flow - Average

Fig. 12. [Large-scale 1 Gbps testbed] FCT results under traffic pattern 1: Inter-DC traffic, highly congested in WAN. Small flow: Size <100 KB. Large flow: Size >10 MB. GEMINI achieves the best or second best results in most cases of Figure 12- 15 (within 11% of the second best scheme in the worst case).



(a) Small Flow - Average (b) Small Flow - 99th Tail (c) Large Flow - Average (d) All Flow - Average

Fig. 13. [Small-scale 10 Gbps testbed] FCT results under traffic pattern 1: Inter-DC traffic, highly congested in WAN. Small flow: Size <100 KB. Large flow: Size >10 MB.

throughput. Reducing β improves throughput but may hurt convergence speed at the same time. We recommend to set $T = 5\,ms$ and $\beta = 0.2$. T that is a bit higher than needed in this case, leaving more room for higher RTT networks. In practice, this is also necessary to filter out the interference from DCN queueing delay (usually within 1 ms). We repeat the previous experiments under higher RTTs with unchanged parameter settings. Results show that GEMINI can still achieve 857 mbps throughput (within 10% of the highest throughput) under 100 ms base RTT. This verifies the default settings can work well under a wide RTT range.

C. Convergence, Stability and Fairness

To evaluate the convergence and stability of GEMINI, we first start a group of 10 flows from one server. At 50 seconds, we start a second group of flows from another server in the same rack. At 100 seconds, we start a third group of flows from another rack in the same DC. All flows run for 150 seconds and share the same destination server in a remote DC.

Figure 10 shows the throughput dynamics (one flow is shown for each flow group). GEMINI guarantees fair convergence given its AIMD nature. In fact, GEMINI converges quickly and stably under both DCN congestion (50–100 secs) and WAN congestion (100–200 secs). For example, during 100–150 secs, the average Jain's fairness index [36] of GEMINI is 0.996, much better than the other protocols (0.926, 0.975, 0.948 for Cubic, DCTCP and BBR, respectively).

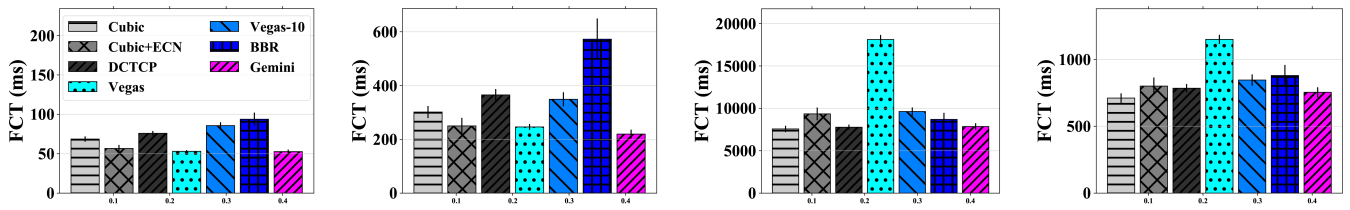
Fairness is important for good tail performance. RTT unfairness [32], [33] is the major challenge in achieving per-flow bandwidth fair-sharing in cross-DC networks, where intra-DC and inter-DC traffic with different RTTs coexists. We show that, good RTT-fairness can be achieved by GEMINI with the factor h and the scale factor F. To demonstrate that, we generate 4 inter-DC flows and 4 intra-DC flows sharing the same bottleneck link inside DC. The intra-DC RTT is $\sim 200\,\mu$s. With tc NETEM [22], the inter-DC RTT is set to $4\times$, $8\times$, $16\times$, $32\times$, $64\times$ the intra-DC RTT. All ECN-enabled protocols adopt the same ECN threshold of 300 packets for fair comparison. The experiment result is shown in Figure 11. While Cubic and DCTCP achieve proportional RTT-fairness and BBR skews towards large RTT flows, GEMINI maintains equal bandwidth fair-sharing regardless of the varying RTTs.

D. Realistic Workloads

We evaluate GEMINI under realistic workloads. The workloads are generated based on traffic patterns that have been observed in a data center supporting web search [10]. Flows arrive by the Poisson process. The source and destination is chosen uniformly random from a configured IP pool. The workload is heavy-tailed with about 50% small flows (size <100 KB) while 80% of all bytes belong to the larger 10% of the flows of size greater than 10 MB. We run the workload with a publicly available traffic generator that has been used by other work [37], [38]. Similar to prior work [10], [39]–[41], we use flow completion time (FCT) as the main performance metric.

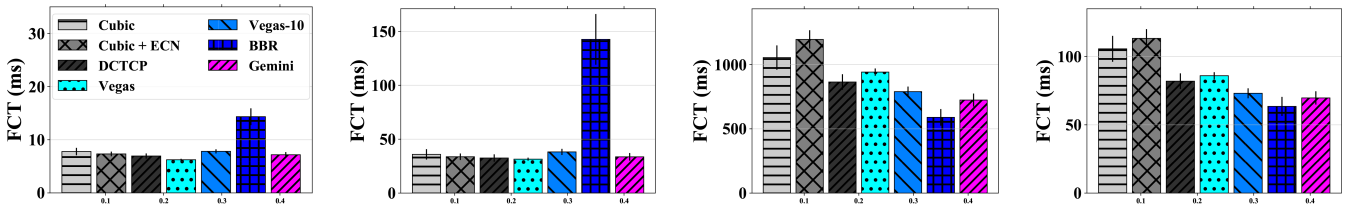
Traffic Pattern 1: Inter-DC Traffic, Highly Congested in WAN: In this experiment, all flows cross the WAN segment. The average utilization of the inter-DC WAN link is $\sim$90%. The DC border routers are highly congested, while intra-DC links have much lower utilization ($\sim$11.25–45%).

The experiment results are shown in Figure 12 and 13: (1) For small flow FCT, GEMINI performs better than Cubic, DCTCP and BBR on both average and 99th tail. This is because Cubic and DCTCP suffer from the large queueing



(a) Small Flow - Average (b) Small Flow - 99th Tail (c) Large Flow - Average (d) All Flow - Average

Fig. 14. [Large-scale 1 Gbps testbed] FCT results under traffic pattern 2: mixed inter-DC and intra-DC traffic, highly congested both in WAN and DCN. Small flow: Size $<$ 100 KB. Large flow: Size $>$ 10 MB.



(a) Small Flow - Average (b) Small Flow - 99th Tail (c) Large Flow - Average (d) All Flow - Average

Fig. 15. [Small-scale 10 Gbps testbed] FCT results under traffic pattern 2: mixed inter-DC and intra-DC traffic, highly congested both in WAN and DCN. Small flow: Size $<$ 100 KB. Large flow: Size $>$ 10 MB.

delay in WAN segment while GEMINI well handles that with RTT signal. BBR suffers a lot from loss as the misestimates of bandwidth and RTT are magnified by high congestion. BBR does not react to loss events explicitly until loss rate $> 20\%$ (as a protection). This design choice benefits the long-term throughput while hurts short-term latency. (2) For large flow FCT, GEMINI performs much better than Vegas. The default parameter setting for Vegas is very conservative ($\alpha = 2, \beta = 4$), leading to poor throughput of large flows. Setting larger thresholds in Vegas-10 ($\alpha = 20, \beta = 40$) improves throughput but hurts latency of small flows. (3) For overall FCT, GEMINI performs the best among all experimented transport protocols.

Traffic Pattern 2: Mixed Traffic, Highly Congested Both in WAN and DCN: In this experiment, the source and the destination of each flow is chosen uniformly random among all servers. Intra-DC and inter-DC traffic coexists in the network. The average utilization of the inter-DC WAN link is $\sim$90%. The average utilization of the link from the DC switch to the border router is $\sim$67.5%. Therefore, both WAN and DCN are highly congested.

The experiment results are shown in Figure 14 and 15: (1) For small flow FCT, GEMINI performs one of the best among experimented transport protocols. In fact, GEMINI has consistently low packet loss rates ($< 10\times10^{-5}$) under both traffic patterns. This is because Gemini handles DCN congestion adaptively with different RTTs, thus allowing a low ECN threshold with larger buffer headroom to absorb burst. Besides, it also enforces pacing to reduce burst losses. (2) For large flow FCT, GEMINI performs better than Cubic and Vegas. Vegas does not perform well because it cannot control congestion in WAN and DCN simultaneously. GEMINI can identify and react to congestion in DCN and WAN differently using ECN and RTT signals respectively. (3) For overall FCT, GEMINI performs one of the best among all experimented transport protocols.

V. DISCUSSION

A. Practical Considerations

TCP Friendliness: GEMINI is not TCP-friendly. For example, like all ECN-based protocols, GEMINI has fairness issues if it coexists with non-ECN protocols. In fact, recent work [15], [42] shows that it is fundamentally difficult to achieve high performance while attaining perfect friendliness to buffer-filling protocols. Thus, they sacrifice performance to guarantee friendliness when detecting the buffer-fillers. GEMINI can adopt similar approaches (*e.g.*, switching to TCP-competitive mode when buffer-fillers are detected). We believe advance switch support like ideal fair queueing [43] would be a better solution. We do not focus on the problem in this paper.

Real-world Deployment: For private clouds, deploying GEMINI requires the cloud owners to add the new congestion control kernel module at the end-hosts and configure ECN at the DC switches. In terms of partial deployment, GEMINI can work with common TCP remote ends since it relies on exactly the same ACK mechanism as TCP Cubic. For public clouds, cloud users and cloud operators are of different entities. On one hand, cloud users control the VMs and thus can deploy GEMINI with minimal support (ECN only) from cloud operators. On the other hand, cloud operators control the hypervisors at the end-hosts and the underlying network devices. GEMINI can be enforced similarly like [44], [45].

B. Alternative Solutions

Multiple Queues + Different Protocols: A seemingly straightforward solution under cross-DC network is to use different transport protocols for intra-DC and inter-DC traffic, like what major infrastructure owners, *e.g.*, Google [14], [19] and Microsoft [10], [46] do for their first-party applications. However, a few issues make this approach less attractive:

(1) classifying inter-DC and intra-DC flows based on IPs is nontrivial. In cloud network, virtual subnets extend across DCs for the ease of management, decoupling the IPs from the DC locations. Maintaining an always-up-to-date IP-to-DC mapping globally is daunting and gives away the management benefit of running virtual subnets; (2) different transport protocols are unlikely to fair-share the network bandwidth, thus requiring the switches to allocate different queues for them. General cloud users usually do not have this luxury; (3) even if we pay the cost to adopt such a solution, some desired performance goals are still missed: (a) The inter-DC traffic may encounter congestion on either WAN or DCN. The existing protocols do not address both at the same time. The inter-DC traffic that tends to use wide-area transport will suffer from low throughput and high loss rate when experiencing congestion at the shallow-buffered DC switch; (b) Flow-level bandwidth fair-sharing cannot be guaranteed by coarse-grained traffic isolation. Static bandwidth allocation may even lead to bandwidth under-utilization.

TCP Proxy: Another possible solution is to terminate and relay the TCP flows with proxies at the border of each network, or the so-called Split TCP [47]–[49]. In this way, traffic can be transported using the best-suited protocol for each network. The TCP proxy way has flaws in practice:

- Proxies in the middle add extra latencies. The latency overhead can greatly impair network performance, especially for short flows that can finish in one RTT.
- Relaying every inter-DC flow is impractical. As a rule of thumb, Google [19] allocates 10% of aggregate intra-DC bandwidth for external connectivity, requiring prohibitive amount of relay bandwidth for peak WAN usage.
- Configuring and managing proxy chains is complex and error-prone. Single fault from one of the intermediate relays may tear down the whole communication.

VI. Related Work

To facilitate cross-DC network communication, there are three lines of work in general, each operating on a different granularity. First, WAN traffic engineering [3], [4], [50] works on the datacenter level. It distributes network traffic to multiple site-to-site paths (usually hundreds of updates per day). Second, bandwidth allocation [51] applies to the tenant or flow group level. It re-allocates the site-to-site bandwidths and split them among all competing flow groups. Third, transport protocol regulates the per-flow sending rate in real-time. We focus on the transport design in this paper.

To the best of our knowledge, transport design under heterogeneous cross-DC network is unexplored in literature. However, there are vast transport protocols under wide area network and datacenter network that are highly related:

Wide Area Network Transport: Cubic [23] is the default TCP congestion control in the Linux system. It achieves high scalability and proportional RTT-fairness by growing window with a cubic function of time. Vegas [12] is the seminal transport protocol that uses delay signal to avoid intrinsic high loss and queueing delay of loss-based transport. After that, many WAN protocols [14], [15], [52] are proposed to use delay signal. For example, BBR [14] is proposed primarily for the enterprise WAN. The core idea is to work at the theoretically optimal point [35] with the aid of sophisticated network sensing (*e.g.*, precise bandwidth and RTT estimation). These transport protocols consider WAN only and usually suffer a lot from the intra-DC congestion in cross-DC network.

Datacenter Network Transport: DCTCP [10] detects the network congestion with ECN and react in proportion to the measured extent of congestion. Following that, many ECN-based protocols [11], [53]–[55] are proposed for DCN congestion control. The other line of work leverages delay signal with microsecond-level accuracy for congestion feedback, which is enabled by recent advances [13], [56], [57] in NIC technology. For example, TIMELY [13] and RoGUE [58] use RTT signal for congestion control in RDMA communication. We show that ECN or delay signal alone is insufficient for cross-DC congestion control.

Explicit rate control [59]–[63] allocates network resources with in-network assistance. Centralized rate control [64], [65] regulates traffic rate with a centralized controller. Multi-path transport [66]–[69] creates subflows along multiple paths to achieve high aggregate throughput and traffic load balance. Proactive congestion control [70]–[75] leverages receiver-side credits to trigger new traffic sending. Learning-based congestion control [76]–[78] learns the congestion control strategy by machine either online or offline. These novel transport protocols often have less comprehensive and unpredictable performance, and may require advanced network support (*e.g.*, cutting payload [79]) that are unavailable or bad supported in cross-DC network facilities.

VII. Conclusion

As geo-distributed applications become prevalent, cross-DC communication gets increasingly important. We investigate existing transport and find that they leverage either ECN or delay signal alone, which cannot accommodate the heterogeneity of cross-DC networks. Motivated by this, we design Gemini, a solution for cross-DC congestion control that integrates both ECN and delay signal. Gemini uses the delay signal to bound the total in-flight traffic end-to-end, while ECN is used to control the per-hop queues inside a DCN. It further modulates ECN-triggered window reduction aggressiveness with RTT to achieve high throughput under limited buffer. We implement Gemini with Linux kernel and commodity switches. Experiments show that Gemini achieves low latency, high throughput, fair and stable convergence, and delivers lower FCTs compared to various transport protocols (*e.g.*, Cubic, Vegas, DCTCP and BBR) in cross-DC networks.

Appendix

A. Derivation of the Scale Factor F

We analyze the steady state behavior and prove that Gemini achieves full throughput with scale factor $F = \frac{4K}{C \times RTT + K}$.

Theorem 1: Given a positive ECN marking threshold K, we can maintain 100% throughput under DCN congestion if congestion window is reduced as follows,

$$CWND = CWND \times (1 - \alpha \times F)$$

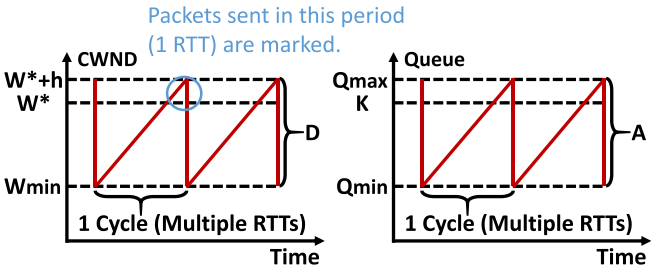


Fig. 16. AIMD Sawtooth illustration.

where α *is the EWMA of ECN fraction and* $F \leq \frac{4K}{C \times RTT + K}$.

Proof: Same as prior work [10], [15], we consider N long-lived flows with identical round-trip times RTT, sharing a single bottleneck link of capacity C. Assuming N window sizes are synchronized for the ease of analysis, the queue size is:

$$Q(t) = N \times W(t) - C \times RTT \tag{1}$$

where $W(t)$ is the dynamic window size. Therefore, the queue size process is also a sawtooth. To achieve full link utilization, we need to guarantee: $Q_{min} \geq 0$ (see Figure 16).

The queue size exceeds the marking threshold K for exactly one RTT in each cycle before the sources receive ECN feedback and reduce their window sizes accordingly. We can compute the fraction of marked packets, α, by simply dividing the number of packets sent during the last RTT of the cycle by the total number of packets sent during a full cycle of the sawtooth.

Let's consider one of the senders. Let $S(W_1, W_2)$ denote the number of packets sent by the sender, while its window size increases from W_1 to $W_2 > W_1$. Since this takes $(W_2 - W_1)/h$ round trip times, during which the average window size is $(W_1 + W_2)/2$,

$$S(W_1, W_2) = (W_2^2 - W_1^2)/2h \tag{2}$$

Let $W^* = (C \times RTT + K)/N$. This is the critical window size at which the queue size reaches K, and the switch starts marking packets with the Congestion Experienced (CE) code-point. During the RTT before the sender reacts, the window size peaks at $W^* + h$. Hence,

$$\alpha = S(W^*, W^* + h)/S((W^* + h)(1 - \alpha F), W^* + h) \tag{3}$$

Plugging (2) into (3) and rearranging, we get:

$$\alpha^2 F(2 - \alpha F) = (2W^* + h)h/(W^* + h)^2 \approx 2h/W^* \tag{4}$$

where the approximation is valid when $W^* \gg h$.

Equation (4) can be used to compute α as a function of the network parameters C, RTT, N and K. Assuming $\alpha F/2$ is small, this can be simplified as:

$$\alpha \approx \sqrt{h/FW^*} \tag{5}$$

We can now compute A in Figure 16 as follows. Note that the amplitude of oscillation in window size of a single flow, D, is given by:

$$D = (W^* + h) - (W^* + h)(1 - \alpha F) = (W^* + h)\alpha F \tag{6}$$

Since there are N flows in total,

$$A = N \times D = N(W^* + h)\alpha F \approx N\sqrt{hFW^*} = \sqrt{NhF(C \times RTT + K)} \tag{7}$$

With (1), we have:

$$Q_{max} = N \times (W^* + h) - C \times RTT = K + Nh \tag{8}$$

With (7) and (8), the minimum queue length is:

$$Q_{min} = Q_{max} - A = K + Nh - \sqrt{NhF(C \times RTT + K)} \tag{9}$$

Finally, to find the relationship between the scale factor F and the ECN marking threshold K, we minimize (9) over N, and choose K and F so that this minimum is no smaller than zero (*i.e.*, the queue never underflows). This results in:

$$F \leq \frac{4K}{C \times RTT + K} \tag{10}$$

As we can see, given a fixed ECN marking threshold K, the larger RTT a flow has, the smaller F it gets. Therefore, the flows with larger RTTs adjust window more smoothly to achieve high throughput.

Note that the theoretical analysis here is a generalized form of that in the DCTCP paper [10] and the result is consistent with it. Specifically, when following the DCTCP algorithm by setting a constant parameter $F = \frac{1}{2}$, we have $K \geq (C \times RTT)/7$, exactly matching the original DCTCP guideline.

B. Proof of RTT-Fairness

We show GEMINI achieves fair-share of the bottleneck bandwidth in DCN where inter-DC and intra-DC flows coexist.

Theorem 2: GEMINI *achieves ideal RTT-fairness with following AIMD rule:*

Decrease: When congestion indicated by ECN per RTT,

$$CWND = CWND \times (1 - \alpha \times F)$$

where α is the ECN fraction and $F = \frac{4K}{C \times RTT + K}$.

Increase: When there is no congestion indication per ACK,

$$CWND = CWND + \frac{h}{CWND}$$

where h is an adaptive congestion avoidance function in proportion to BDP: $h \propto RTT$.

Proof: From previous subsection, we know that the average window size is:

$$\overline{W} = \frac{W^* + h + (W^* + h) \times (1 - \alpha F)}{2} \tag{11}$$

Therefore, when two flows competing for one bottleneck link reach the steady state:

$$\frac{\overline{W_1}}{\overline{W_2}} = \frac{(W_1^* + h_1) \times (1 - \frac{\alpha_1 F_1}{2})}{(W_2^* + h_2) \times (1 - \frac{\alpha_2 F_2}{2})} \approx \frac{W_1^*}{W_2^*} \tag{12}$$

when assuming that $1 \gg \frac{\alpha F}{2}$ and $W^* \gg h$.

When two flows F1 and F2 with different RTTs (assuming $RTT_1 < RTT_2$) are competing on one bottleneck link, Equation 5 is still valid for the small RTT flow F1, in other form:

$$W_1^* = 2h_1/(F_1\alpha_1^2) \tag{13}$$

However, Equation 13 does not hold for large RTT flow F2. When small RTT flow F1 reduces its CWND as soon as it gets the ECN feedback after RTT_1, the bottleneck queue length drops immediately and packets of large RTT flow F2 will stop being marked with ECN. So flow F2 will get only around $S(W_2^*, W_2^* + h_2)\frac{RTT_1}{RTT_2}$ packets marked with ECN. Following same approach from Equation 3 to 13, for F2,

$$W_2^* = 2h_2/(F_2\alpha_2^2) \times \frac{RTT_1}{RTT_2} \tag{14}$$

Packets traversing the same link have the same probability to be ECN marked. Thus, we get:

$$\alpha_1 = \alpha_2 \tag{15}$$

Plugging Equation 10, 13, 14, 15 into Equation 12, we have:

$$\frac{\overline{W_1}}{\overline{W_2}} = \frac{F_2}{F_1} = \frac{C \times RTT_1 + K}{C \times RTT_2 + K} \tag{16}$$

When assuming the average queue length is around K. We have the average RTT:

$$\overline{RTT} \approx RTT + \frac{K}{C} \tag{17}$$

Therefore, we have the bandwidth sharing ratio:

$$\frac{\overline{R_1}}{\overline{R_2}} = \frac{\overline{W_1}}{\overline{RTT_1}} / \frac{\overline{W_2}}{\overline{RTT_2}} \approx 1 \tag{18}$$

where R_i denotes the sending rate of flow i.

REFERENCES

[1] G. Zeng *et al.*, "Congestion control for cross-datacenter networks," in *Proc. IEEE 27th Int. Conf. Netw. Protocols (ICNP)*, Oct. 2019, pp. 1–12.

[2] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez, "Inter-datacenter bulk transfers with netstitcher," in *Proc. ACM SIGCOMM Conf. SIGCOMM (SIGCOMM)*, 2011, pp. 74–85.

[3] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined wan," in *Proc. ACM SIGCOMM Conf. (SIGCOMM)*, Aug. 2013, pp. 3–14.

[4] C.-Y. Hong *et al.*, "Achieving high utilization with software-driven WAN," in *Proc. ACM SIGCOMM Conf. SIGCOMM*, Aug. 2013, pp. 15–26.

[5] Z. Wu, M. Butkiewicz, D. Perkins, E. Katz-Bassett, and H. V. Madhyastha, "SPANStore: Cost-effective geo-replicated storage spanning multiple cloud services," in *Proc. 24th ACM Symp. Operating Syst. Princ.*, Nov. 2013, pp. 292–308.

[6] Q. Pu *et al.*, "Low latency geo-distributed data analytics," in *Proc. ACM Conf. Special Interest Group Data Commun.*, Aug. 2015, pp. 421–434.

[7] C.-C. Hung, L. Golubchik, and M. Yu, "Scheduling jobs across geo-distributed datacenters," in *Proc. 6th ACM Symp. Cloud Comput.*, Aug. 2015, pp. 111–124.

[8] X. Jin *et al.*, "Optimizing bulk transfers with software-defined optical WAN," in *Proc. ACM SIGCOMM Conf.*, Aug. 2016, pp. 87–100.

[9] K. Hsieh *et al.*, "Gaia: Geo-distributed machine learning approaching LAN speeds," in *Proc. NSDI*, 2017, pp. 629–647.

[10] M. Alizadeh *et al.*, "Data center TCP (DCTCP)," in *Proc. ACM SIGCOMM Conf. SIGCOMM (SIGCOMM)*, 2010, pp. 63–74.

[11] Y. Zhu *et al.*, "Congestion control for large-scale RDMA deployments," in *Proc. ACM Conf. Special Interest Group Data Commun.*, Aug. 2015, pp. 523–536.

[12] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," in *Proc. SIGCOMM*, 1994, pp. 24–35.

[13] R. Mittal *et al.*, "TIMELY: RTT-based congestion control for the datacenter," in *Proc. ACM Conf. Special Interest Group Data Commun.*, Aug. 2015, pp. 537–550.

[14] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control," in *Proc. ACM Queue*, 2016, pp. 58–66.

[15] V. Arun and H. Balakrishnan, "Copa: Practical delay-based congestion control for the internet," in *Proc. NSDI*, 2018, pp. 329–342.

[16] G. Zeng, S. Hu, J. Zhang, and K. Chen, "Transport protocols for data center networks: A survey," *J. Comput. Res. Develop.*, vol. 57, no. 1, p. 74, 2020.

[17] *A. Production*. Accessed: 2019. [Online]. Available: https://www.arista.com/en/products

[18] 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.*, Aug. 2015, pp. 123–137.

[19] A. Singh *et al.*, "Jupiter rising: A decade of clos topologies and centralized control in Google's datacenter network," in *Proc. ACM Conf. Special Interest Group Data Commun.*, Aug. 2015, pp. 183–197.

[20] S. Bauer, R. Beverly, and A. Berger, "Measuring the state of ECN readiness in servers, clients,and routers," in *Proc. ACM SIGCOMM Conf. Internet Meas. Conf. (IMC)*, 2011, pp. 171–180.

[21] M. Kuhlewind, D. P. Wagner, J. M. R. Espinosa, and B. Briscoe, "Using data center TCP (DCTCP) in the internet," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2014, pp. 583–588.

[22] *Netem in Linux Foundation Wiki*. Accessed: 2019. [Online]. Available: https://wiki.linuxfoundation.org/networking/netem

[23] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," in *Proc. SIGOPS*, 2008, pp. 64–74.

[24] K. Ramakrishnan and R. Jain, "A binary feedback scheme for congestion avoidance in computer networks," in *Proc. TOCS*, 1990, pp. 158–181.

[25] K. Ramakrishnan, S. Floyd, and D. Black, *The Addition of Explicit Congestion Notification (ECN) to IP*, document RFC 3168, 2001.

[26] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," in *Proc. Conf. Appl., Technol., Archit., Protocols Comput. Commun. (SIGCOMM)*, 2004, pp. 281–292.

[27] M. Alizadeh, A. Javanmard, and B. Prabhakar, "Analysis of DCTCP: Stability, convergence, and fairness," in *Proc. ACM SIGMETRICS Joint Int. Conf. Meas. Modeling Comput. Syst. (SIGMETRICS)*, 2011, pp. 73–84.

[28] J. Corbet. *TSO sizing FQ scheduler*. Accessed: 2019. [Online]. Available: https://lwn.net/Articles/564978/

[29] G. Zeng, W. Bai, G. Chen, K. Chen, D. Han, and Y. Zhu, "Combining ECN and RTT for datacenter transport," in *Proc. 1st Asia–Pacific Workshop Netw.*, Aug. 2017, pp. 36–42.

[30] K. Nichols and V. Jacobson, "Controlling queue delay," in *Proc. ACM Queue*, 2012, pp. 42–50.

[31] V. Jacobson, "Congestion avoidance and control," in *Proc. Symp. Commun. Archit. Protocols (SIGCOMM)*, 1988, pp. 314–329.

[32] T. V. Lakshman and U. Madhow, "The performance of TCP/IP for networks with high bandwidth-delay products and random loss," *IEEE/ACM Trans. Netw.*, vol. 5, no. 3, pp. 336–350, Jun. 1997.

[33] P. Brown, "Resource sharing of TCP connections with different round trip times," in *Proc. IEEE INFOCOM. Conf. Comput. Commun. 19th Annu. Joint Conf. IEEE Comput. Commun. Societies*, Mar. 2000, pp. 1734–1741.

[34] A. K. Choudhury and E. L. Hahne, "Dynamic queue length thresholds for shared-memory packet switches," *IEEE/ACM Trans. Netw.*, vol. 6, no. 2, pp. 130–140, Apr. 1998.

[35] L. Kleinrock, "Power and deterministic rules of thumb for probabilistic problems in computer communications," in *Proc. ICC*, 1979, p. 43.

[36] R. K. Jain, D.-M. W. Chiu, and W. R. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," in *Research Report of Digital Equipment Corporation*, 1984. [Online]. Available: https://arxiv.org/abs/cs/9809099

[37] B. Li *et al.*, "ClickNP: Highly flexible and high performance network processing with reconfigurable hardware," in *Proc. ACM SIGCOMM Conf.*, Aug. 2016, pp. 1–14.

[38] W. Bai, L. Chen, K. Chen, and H. Wu, "Enabling ECN in multi-service multi-queue data centers," in *Proc. NSDI*, 2016, pp. 537–549.

[39] M. Alizadeh *et al.*, "PFabric: Minimal near-optimal datacenter transport," in *Proc. ACM SIGCOMM Conf. (SIGCOMM)*, Aug. 2013, pp. 435–446.

[40] J. Xia *et al.*, "Rethinking transport layer design for distributed machine learning," in *Proc. 3rd Asia–Pacific Workshop Netw.*, Aug. 2019, pp. 22–28.

[41] H. Wang *et al.*, "Domain-specific communication optimization for distributed DNN training," 2020, *arXiv:2008.08445*.

[42] P. Goyal, A. Narayan, F. Cangialosi, S. Narayana, M. Alizadeh, and H. Balakrishnan, "Elasticity detection: A building block for internet congestion control," 2018, *arXiv:1802.08730*.

[43] N. K. Sharma, M. Liu, K. Atreya, and A. Krishnamurthy, "Approximating fair queueing on reconfigurable switches," in *Proc. NSDI*, 2018, pp. 1–16.

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

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

[46] C.-Y. Hong *et al.*, "Achieving high utilization with software-driven WAN," in *Proc. ACM SIGCOMM Conf. (SIGCOMM)*, Aug. 2013, pp. 15–26.

[47] S. Kopparty, S. V. Krishnamurthy, M. Faloutsos, and S. K. Tripathi, "Split TCP for mobile ad hoc networks," in *Proc. Global Telecommun. Conf. (GLOBECOM)*, Nov. 2002, pp. 138–142.

[48] T. Flach *et al.*, "Reducing web latency: The virtue of gentle aggression," in *Proc. ACM SIGCOMM Conf. (SIGCOMM)*, Aug. 2013, pp. 159–170.

[49] F. Le, E. Nahum, V. Pappas, M. Touma, and D. Verma, "Experiences deploying a transparent split TCP middlebox and the implications for NFV," in *Proc. ACM SIGCOMM Workshop Hot Topics Middleboxes Netw. Function Virtualization*, Aug. 2015, pp. 31–36.

[50] P. Kumar *et al.*, "Semi-oblivious traffic engineering: The road not taken," in *Proc. NSDI*, 2018, pp. 157–170.

[51] A. Kumar *et al.*, "BwE: Flexible, hierarchical bandwidth allocation for WAN distributed computing," in *Proc. ACM Conf. Special Interest Group Data Commun.*, Aug. 2015, pp. 1–14.

[52] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A compound TCP approach for high-speed and long distance networks," in *Proc. IEEE INFOCOM 25th Int. Conf. Comput. Commun.*, Apr. 2006, pp. 1–12.

[53] M. Alizadeh, A. Kabbani, T. Eds.,all, B. Prabhakar, A. Vahdat, and M. Yasuda, "Less is more: Trading a little bandwidth for ultra-low latency in the data center," in *Proc. NSDI*, 2012, pp. 253–266.

[54] B. Vamanan, J. Hasan, and T. N. Vijaykumar, "Deadline-aware datacenter TCP (D2TCP)," in *Proc. ACM SIGCOMM Conf. Appl., Technol., Archit., Protocols Comput. Commun. (SIGCOMM)*, 2012, pp. 115–126.

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

[56] S. Han, K. Jang, A. Panda, S. Palkar, D. Han, and S. Ratnasamy, "SoftNIC: A software NIC to augment hardware," EECS Dept., Univ. California, Berkeley, CA, USA, Tech. Rep., UCB/EECS-2015-155, 2015.

[57] C. Lee, C. Park, K. Jang, S. Moon, and D. Han, "Dx: Latency-based congestion control for datacenters," *IEEE/ACM Trans. Netw.*, vol. 25, no. 1, pp. 335–348, Feb. 2017.

[58] Y. Le, B. Stephens, A. Singhvi, A. Akella, and M. Swift, "RoGUE: RDMA over generic unconverged Ethernet," in *Proc. SoCC*, 2018, pp. 225–236.

[59] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *Proc. Conf. Appl., Technol., Archit., Protocols Comput. Commun. (SIGCOMM)*, 2002, pp. 89–102.

[60] N. Dukkipati, M. Kobayashi, R. Zhang-Shen, and N. McKeown, "Processor sharing flows in the internet," in *Proc. IWQoS*, 2005, pp. 271–285.

[61] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron, "Better never than late: Meeting deadlines in datacenter networks," in *Proc. ACM SIGCOMM Conf. SIGCOMM (SIGCOMM)*, 2011, pp. 50–61.

[62] C.-Y. Hong, M. Caesar, and P. B. Godfrey, "Finishing flows quickly with preemptive scheduling," in *Proc. ACM SIGCOMM Conf. Appl., Technol., Archit., Protocols Comput. Commun. (SIGCOMM)*, 2012, pp. 127–138.

[63] D. Han, R. Grandl, A. Akella, and S. Seshan, "FCP: A flexible transport framework for accommodating diversity," in *Proc. ACM SIGCOMM Conf. (SIGCOMM)*, Aug. 2013, pp. 135–146.

[64] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal, "Fastpass: A centralized 'zero-queue' datacenter network," in *Proc. ACM Conf. (SIGCOMM)*, Aug. 2014, pp. 307–318.

[65] J. Perry, H. Balakrishnan, and D. Shah, "Flowtune: Flowlet control for datacenter networks," in *Proc. NSDI*, 2017, pp. 421–435.

[66] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in *Proc. NSDI*, 2011, pp. 1–14.

[67] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath TCP," in *Proc. ACM SIGCOMM Conf. SIGCOMM (SIGCOMM)*, 2011, pp. 266–277.

[68] Y. Lu *et al.*, "Multi-path transport for RDMA in datacenters," in *Proc. NSDI*, 2018, pp. 357–371.

[69] G. Zeng, L. Chen, B. Yi, and K. Chen, "Cutting tail latency in commodity datacenters with cloudburst," in *Proc. IEEE INFOCOM*, Jan. 2022, pp. 1–10.

[70] I. Cho, K. Jang, and D. Han, "Credit-scheduled delay-bounded congestion control for datacenters," in *Proc. Conf. ACM Special Interest Group Data Commun.*, Aug. 2017, pp. 239–252.

[71] 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.*, Aug. 2017, pp. 29–42.

[72] B. Montazeri, Y. Li, M. Alizadeh, and J. Ousterhout, "Homa: A receiver-driven low-latency transport protocol using network priorities," in *Proc. Conf. ACM Special Interest Group Data Commun.*, Aug. 2018, pp. 221–235.

[73] S. Hu *et al.*, "Aeolus: A building block for proactive transport in datacenters," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Archit., Protocols Comput. Commun.*, Jul. 2020, pp. 422–434.

[74] S. Hu *et al.*, "Aeolus: A building block for proactive transport in datacenter networks," *IEEE/ACM Trans. Netw.*, early access, Nov. 13, 2021, doi: 10.1109/TNET.2021.3119986.

[75] G. Zeng, J. Qiu, Y. Yuan, H. Liu, and K. Chen, "FlashPass: Proactive congestion control for shallow-buffered WAN," in *Proc. IEEE 29th Int. Conf. Netw. Protocols (ICNP)*, Nov. 2021, pp. 1–12.

[76] K. Winstein and H. Balakrishnan, "TCP ex machina: Computer-generated congestion control," in *Proc. ACM SIGCOMM Conf. (SIGCOMM)*, Aug. 2013, pp. 123–134.

[77] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira, "PCC: Re-architecting congestion control for consistent high performance," in *Proc. NSDI*, 2015, pp. 395–408.

[78] M. Dong *et al.*, "PCC vivace: Online-learning congestion control," in *Proc. NSDI*, 2018, pp. 343–356.

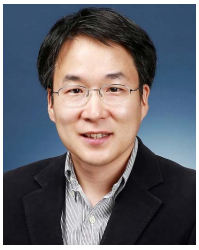
[79] P. Cheng, F. Ren, R. Shu, and C. Lin, "Catch the whole lot in an action: Rapid precise packet loss notification in data center," in *Proc. NSDI*, 2014, pp. 17–28.

Gaoxiong Zeng received the B.E. degree in electronic engineering from the University of Science and Technology of China in 2015 and the Ph.D. degree in computer science and engineering from The Hong Kong University of Science and Technology in 2022. He is a Researcher with the Network Technology Laboratory, Huawei. His research interests include computer networks and systems, with special focuses on data center networking and transport protocols.

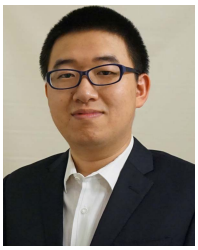
Wei Bai received the B.E. degree in information security from Shanghai Jiao Tong University in 2013 and the Ph.D. degree from the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, in 2017. He is a Senior Researcher with Microsoft Research Laboratory, Redmond. He is broadly interested in computer networking with a special focus on data center networking. Currently, he is mainly focusing on networks infrastructure to enable cloud-scale RDMA deployments.

Ge Chen received the B.S. degree from Shanghai Jiao Tong University in 2013 and the M.S. degree from The Hong Kong University of Science and Technology in 2020. He is currently a Software Engineer with ByteDance, Shanghai.

Kai Chen (Senior Member, IEEE) received the Ph.D. degree in computer science from Northwestern University, Evanston, IL, USA, in 2012. He is an Associate Professor with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong. His current research interests include data center networking, machine learning systems, and privacy-preserving computing.



Dongsu Han (Member, IEEE) received the B.S. degree in computer science from the Korea Advanced Institute of Science and Technology (KAIST) in 2003 and the Ph.D. degree in computer science from Carnegie Mellon University in 2012. He is currently an Associate Professor with the School of Electrical Engineering, KAIST. His research interests include networking, distributed systems, and networks/systems security. More details about his research can be found at http://ina.kaist.ac.kr.



Yibo Zhu received the bachelor's degree from Tsinghua University and the Ph.D. degree from the Department of Computer Science, UCSB, co-advised by Prof. Ben Y. Zhao and Prof. Heather Zheng. He is the Research and Engineering Manager at ByteDance Applied Machine Learning (AML) and the AI Laboratory. Before he joined ByteDance, he was a Senior Researcher at Microsoft Research Laboratory, Redmond. His research interests include distributed machine learning systems and datacenter networks. He builds large-scale software systems using emerging hardware, such as RDMA NICs, GPUs, and programmable ASICs.

Lei Cui is a Researcher with the High Performance Computing Network Laboratory, Huawei. Currently, he focuses on the new computing architecture for the AI/HPC clusters, and the new networks and bus protocols for the architecture.