Congestion Control for Cross-Datacenter Networks

Gaoxiong Zeng
HKUST

Wei Bai
Microsoft

Ge Chen
HKUST

Kai Chen
HKUST

Dongsu Han
KAIST

Yibo Zhu
ByteDance

Lei Cui
Huawei

***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 imposes 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 transports 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, can simultaneously capture the location and degree of congestion. This is 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% and 76% reduction of small flow average completion times compared to TCP Cubic, DCTCP and BBR; and up to 58% reduction of large flow average completion times compared to TCP Vegas.

I. Introduction

Applications running in geographically distributed setting are becoming prevalent [1]–[8]. 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.

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 [9] and DCQCN [10]) or delay (*e.g.*, Vegas [11] and TIMELY [12]) as the congestion signal, and successfully delivered compelling performance in terms of high-throughput and low-latency [9]–[14]. 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.

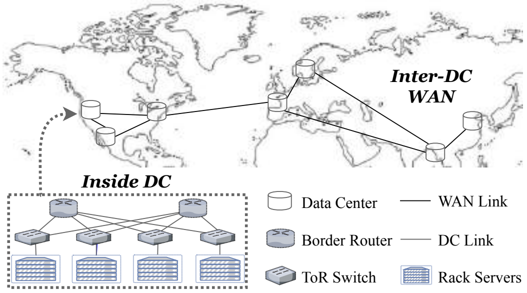


Fig. 1. Cross-Datacenter Network.

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

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× higher throughput at the cost of $>30\times$ more intra-DC packet losses. Furthermore, low delay thresholds impose harsh requirements on accurate delay measurement [12], 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

978-1-7281-2700-2/19/$31.00 2019 © IEEE

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

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 traffics.

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 low flow completion times (FCT)—up to 53%, 31% and 76% reduction of small flow average FCT compared to Cubic, DCTCP and BBR; and up to 58% reduction of large flow average FCT compared to 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:

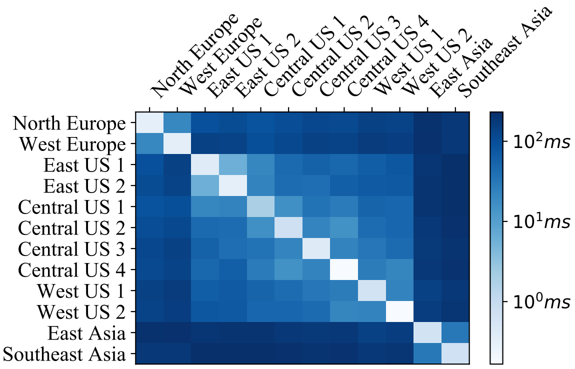


Fig. 2. RTT Heat Map in Cross-DC Network.

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 [15] 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 traffics. Intra-DC and inter-DC traffics coexist in the cross-DC network [16], [17]. 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.

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 [18], [19] (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 transports [9]–[14] 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

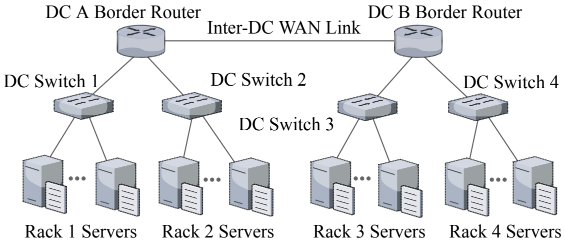


Fig. 3. Cross-Datacenter Network Testbed.

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: We experiment Cubic [21], Vegas [11], BBR [13] and DCTCP [9]. Cubic is experimented with and without ECN. ECN threshold at DC switches is set to 300 packets to guarantee high throughput for inter-DC traffic. 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 [9]. 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:

- Transports 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 traffics*.
- Transports 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 are difficult to achieve under dynamic workload.

Problems of ECN-signal-only solutions. ECN-based transports use the ECN signal [22], [23] that reflects the per-hop network congestion. For it to deliver high throughput, ECN marking threshold must be set proportional to the bandwidth-delay product (BDP) [9], [24], [25]. *However, in a cross-DC setting, it is difficult to configure due to the large difference in RTT 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.

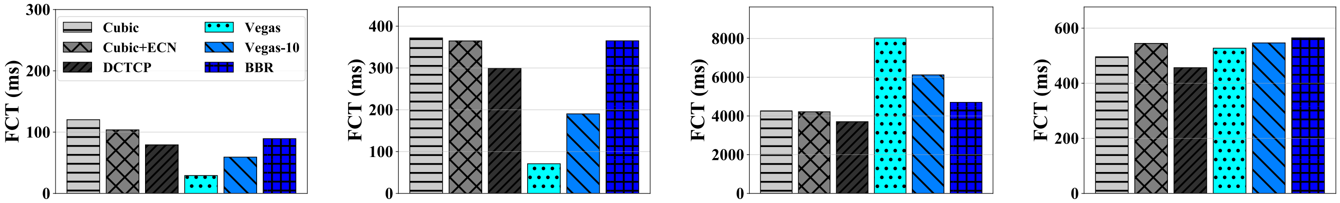
In fact, achieving high utilization over cross-DC is non-trivial because intra-DC switches have shallow buffers — the shallow buffer is easily overwhelmed by bursty large-BDP cross-DC flows (we call it *buffer mismatch*). We confirm that by measuring the packet loss rate (PLR) in previous dynamic workload 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 traffics are now with low BDPs. We observe small PLRs ($< 10 \times 10^{-5}$) 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 [26] 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

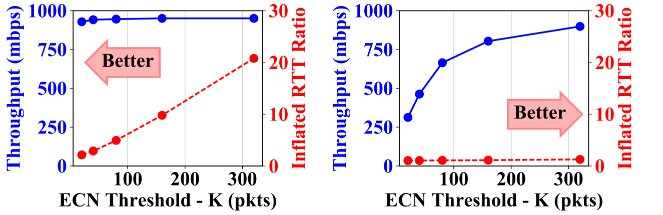
In addition, ECN-based transports require ECN marking support from all network switches. However, ECN marking may not be well supported in WAN. As a result, ECN-based transports may fall back on using packet loss signal, leading to high packet losses and long queueing delay.

[1] Our DC border routers are emulated by servers with multiple NICs, so that we can use NETEM [20] to emulate inter-DC propagation delay.



(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).

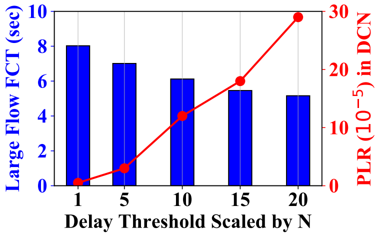


Fig. 6. Dilemma in setting delay threshold. The left y-axis shows throughput by the flow completion time (FCT) of large flows. The right y-axis shows packet loss rate (PLR) inside DCN.

Problems of delay-signal-only solutions. Delay-based transports use the delay signal [11], [12] 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 × 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 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 [12]) are needed.

III. Gemini

We introduce our design rationale in §III-A, describe the detailed Gemini congestion control algorithm in §III-B, and provide theoretical analysis in §III-C and §III-D.

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 transports is naturally resolved.

How to maintain high throughput for inter-DC traffic with shallow-buffered DC switches? A majority of transports (*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 ($C \times RTT$) [9], [24], [25]. 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

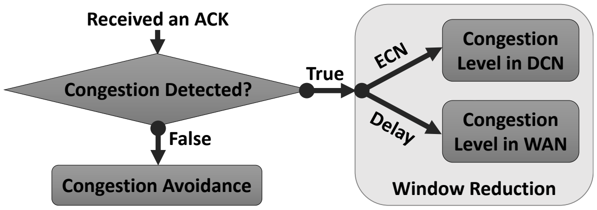


Fig. 7. GEMINI Congestion Control Process.

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.

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. Specifically, DCN congestion is detected by ECN, so as to meet the stringent per-hop queueing control requirement imposed by shallow buffers. WAN congestion is detected by delay, because the end-to-end delay is dominated mostly in WAN than in DCN.

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) codepoint 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.

Algorithm 1: GEMINI Congestion Control Algorithm.

```
Input  : New Incoming ACK
Output: New Congestion Window Size
/* Update transport states: (e.g., $\alpha$).                  */
update_transport_state($\alpha$, 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  ← $\alpha$ × F × congested_dcn ;
        f_wan  ← $\beta$ × 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
α	Average fraction of ECN marked packets
F	Scale factor for DCN congestion control
h	Adaptive congestion window increase step

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 [27], 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 $O(\sqrt{C \times RTT})$ [9], [25]). 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 with detailed proof in §III-C), where C is the bandwidth capacity, 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 $K = (C \times RTT)/7$, 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 [21], [28] do, is enough for high throughput. There are potentially a wide range of threshold settings to effectively work with (see §III-D). 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.

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. Adjusting the window increasing step in proportion to BDP compensates the RTT variation and makes the system more robust under heterogeneity. Further, it also mitigates RTT unfairness [29], [30], 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: $h = H \times C \times RTT$, where H is a constant parameter, C is the bandwidth capacity, RTT is the minimum RTT observed during a long time.

C. 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,*

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

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

Proof: Same as prior work [9], [14], 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:

$$Q(t) = N \times W(t) - C \times RTT \qquad (1)$$

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 8).

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 $(W_1 + W_2)/2$,

$$S(W_1, W_2) = (W_2^2 - W_1^2)/2h \qquad (2)$$

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) codepoint. During the RTT before the sender reacts, the window size peaks at $W^* + h$. Hence,

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

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

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

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

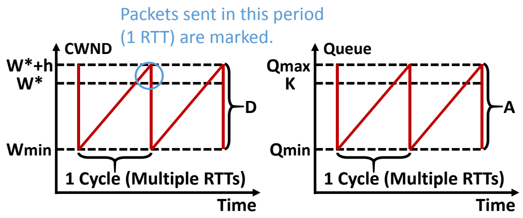


Fig. 8. AIMD Sawtooth Illustration.

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

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

We can now compute A in Figure 8 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,

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

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:

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

D. 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.

TABLE IV
DEFAULT GEMINI PARAMETER SETTINGS.

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

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, we use a per-flow rate limiter at the sender to evenly pace every 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 $T = 5\,ms$.

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 $T > \frac{\beta}{1-\beta}RTT$, or $\beta < \frac{T}{T+RTT}$ in theory (proof similar to [24]). We recommend to set $\beta = 0.2$. We show that a wide range of T and β settings can well serve the cross-DC networks in §IV.

Window Increase Parameter (H). GEMINI additively increases the window by h per RTT when there is no congestion. 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.

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×) and equally low delay compared to DCTCP under DCN congestion; lower delay (>7×) 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 best or the second 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*().

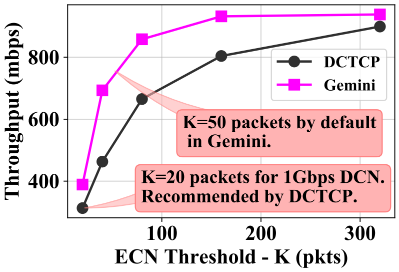


Fig. 9. Aggregate throughput of inter-DC flows that bottlenecked at a DCN link. GEMINI is less buffer-hungry (requires 0-76% smaller K) than DCTCP when achieving similar throughput.

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 [20] 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 [31] 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 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 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 [21], Vegas [11], BBR [13], DCTCP [9] and GEMINI. All these protocols have implementations in Linux kernel and are readily deployable. *Cubic* is the default loss-based congestion control used in Linux system. It is experimented with and without ECN. *DCTCP* is an ECN-based congestion control 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 setting ($\alpha = 20, \beta = 40$) to show the throughput and latency trade-off. *BBR* aims to drive the congestion control to the theoretical optimal point [32] with maximized throughput and minimized latency, based on accurate bandwidth and RTT probing. GEMINI is the transport protocol proposed in this paper. We adopt the default setting (Table IV) throughout all experiments in this paper if not specified. The ECN marking is configured within DCN.

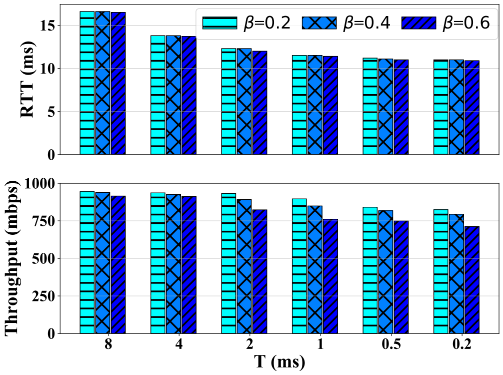


Fig. 10. RTT and throughput of inter-DC flows that bottlenecked at a WAN link. GEMINI achieves equally high throughput (944 mbps) with much lower latency ($> 7\times$) compared to TCP Cubic (> 100 ms).

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 our 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 9. GEMINI maintains slightly higher throughput (938 mbps) than DCTCP (899 mbps) when setting K as high as 320 packets. Setting 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 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 10. In general, GEMINI maintains near full throughput with queueing-delayed RTTs no more than 20 ms. Compared to the transports that leverage loss signals in WAN, GEMINI achieves similar high throughput at the cost of much lower latency (*e.g.*, by $> 7\times$ compared to Cubic that suffers from > 100 ms queueing delay).

Parameter Sensitivity. GEMINI works well under a wide range of parameter settings. To achieve low latency, DCTCP recommends to set ECN marking threshold as low as 20 packets for 1 Gbps DCN. This low ECN threshold significantly

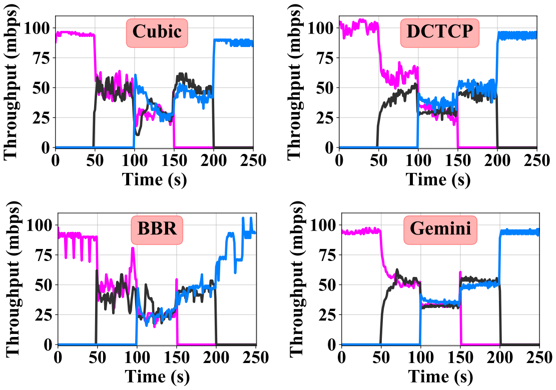


Fig. 11. GEMINI converges quickly, fairly and stably.

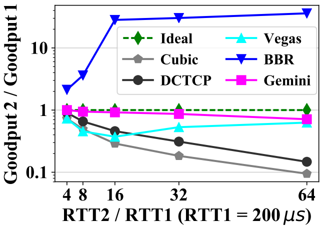


Fig. 12. RTT-fairness. RTT1 and RTT2 are the intra-DC and inter-DC RTTs respectively. GEMINI achieves much better bandwidth fair-sharing.

hurts throughput of DCTCP for large-BDP inter-DC traffic (70% degradation in our experiment result). In fact, DCTCP starts to degrade the throughput when K is set to lower than 300 packets. For GEMINI, the throughput is not degraded until the threshold is set to 100 packets. A higher T leads to higher throughput at the cost of slightly higher RTT. Lower β results in better throughput but sacrifices convergence speed. We recommend to set $T = 5\,ms$ and $\beta = 0.2$. T is a little 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 < 1 ms).

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 11 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 [33] of GEMINI is 0.99, much better than the other protocols.

Fairness is important for good tail performance. RTT unfairness [29], [30] is the major challenge in achieving per-flow bandwidth fair-sharing in cross-DC networks, where intra-DC and inter-DC traffics with different RTTs coexist. 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 [20], the inter-DC RTT is set to 4×, 8×, 16×, 32×, 64× 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 12. 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

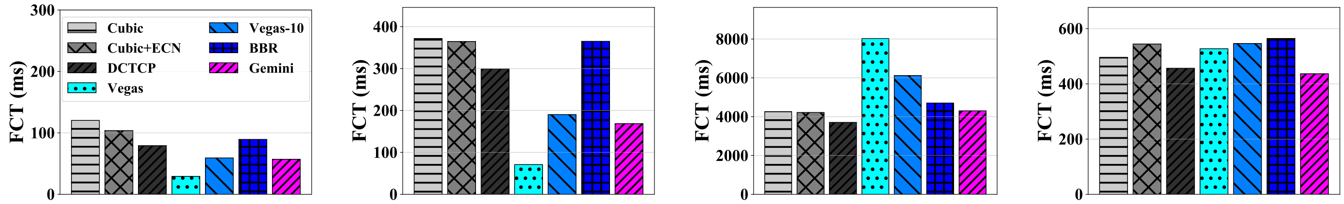
We evaluate GEMINI under realistic workloads. The workloads are generated based on traffic patterns that have been observed in a data center supporting web search [9]. Flows arrive by the Poisson process. The source and destination is chosen uniformly random from a configured IP pool. The workload is heavy-tailed with about 50% small flows (size < 100 KB) while 80% of all bytes belong to the 10% large flows (size > 10 MB). We run the workload with a publicly available traffic generator that has been used by other work [34], [35]. Similar to prior work [9], [36], 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 ∼90%. The DC border routers are highly congested, while intra-DC links have much lower utilization (∼11.25–45%).

The experiment results are shown in Figure 13 and 14: (1) For small flow FCT, GEMINI performs better than Cubic, DCTCP and BBR on both average and 99th tail. Cubic and DCTCP suffer from the large queueing 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 transports.

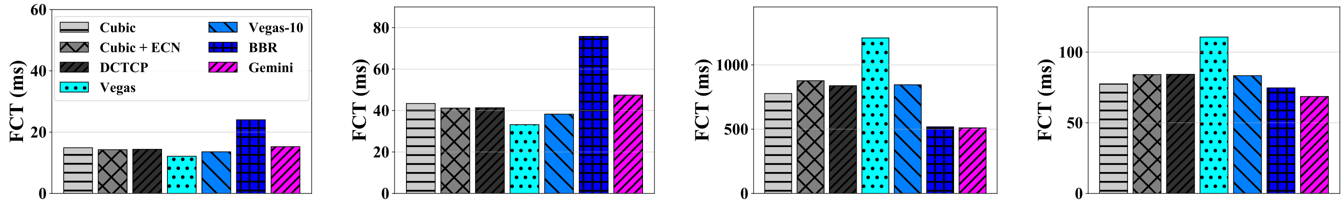
Traffic Pattern 2: Mixed traffic, highly congested both in WAN and DCN. In this experiment, the source and the destination of each flow is chosen uniformly random among all servers. Intra-DC and inter-DC traffics coexist in the network. The average utilization of the inter-DC WAN link is ∼90%. The average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

The experiment results are shown in Figure 15 and 16: (1) For small flow FCT, GEMINI performs one of the best among experimented transports. In fact, GEMINI has consistently low packet loss rates ($< 10 \times 10^{-5}$) under both traffic patterns. BBR suffers from high losses ($> 0.1\%$) again. (2) For large flow FCT, GEMINI performs better than Cubic and Vegas. Vegas



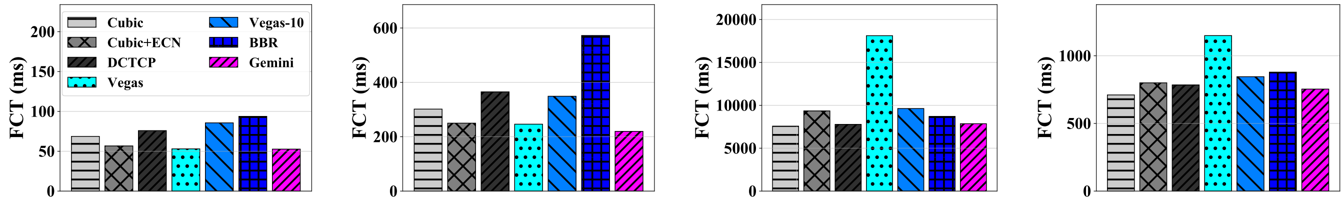
(a) Small Flow - Average (b) Small Flow - 99th Tail (c) Large Flow - Average (d) All Flow - Average

Fig. 13. [Large-scale 1Gbps 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 every case of Figure 13-16, while other protocols have performance hits in certain cases.



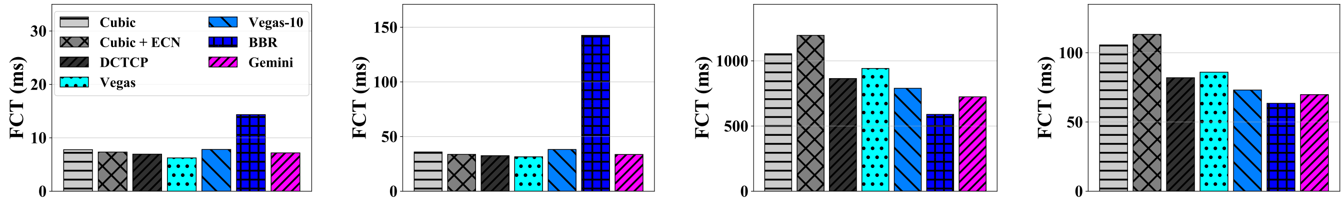
(a) Small Flow - Average (b) Small Flow - 99th Tail (c) Large Flow - Average (d) All Flow - Average

Fig. 14. [Small-scale 10Gbps Testbed] FCT results under traffic pattern 1: Inter-DC traffic, highly congested in WAN.



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

Fig. 15. [Large-scale 1Gbps Testbed] FCT results under traffic pattern 2: mixed inter-DC and intra-DC traffic, highly congested both in WAN and DCN.



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

Fig. 16. [Small-scale 10Gbps Testbed] FCT results under traffic pattern 2: mixed inter-DC and intra-DC traffic, highly congested both in WAN and DCN.

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 transports.

V. Related Work

Wide Area Network Transports. Cubic [21] 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 [11] is the seminal transport protocol that uses delay signal to avoid intrinsic high loss and queueing delay of loss-based transports. After that, many WAN transports [13], [14], [45], [46] are proposed to use delay signal. These transports consider WAN only and usually suffer a lot from the intra-DC congestion in cross-DCN.

Datacenter Network Transports. DCTCP [9] detects network congestion with ECN and react in proportion to the measured extent of congestion. Following that, many ECN-based transports [10], [47]–[49] 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 [12], [50], [51] in NIC technology. For example, TIMELY [12] and RoGUE [52] use delay signal for RDMA congestion control. We show that ECN or delay signal alone is insufficient for cross-DC congestion control.

VI. Conclusion

As geo-distributed applications become prevalent, cross-DC communication gets increasingly important. We investigate existing transports 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. 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.

References

[1] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez, "Inter-datacenter bulk transfers with netstitcher," in *SIGCOMM*, 2011.

[2] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined wan," in *SIGCOMM*, 2013.

[3] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," in *SIGCOMM*, 2013.

[4] Z. Wu, M. Butkiewicz, D. Perkins, E. Katz-Bassett, and H. V. Madhyastha, "Spanstore: Cost-effective geo-replicated storage spanning multiple cloud services," in *SOSP*, 2013.

[5] Q. Pu, G. Ananthanarayanan, P. Bodik, S. Kandula, A. Akella, P. Bahl, and I. Stoica, "Low latency geo-distributed data analytics," in *SIGCOMM*, 2015.

[6] C.-C. Hung, L. Golubchik, and M. Yu, "Scheduling jobs across geo-distributed datacenters," in *SoCC*, 2015.

[7] X. Jin, Y. Li, D. Wei, S. Li, J. Gao, L. Xu, G. Li, W. Xu, and J. Rexford, "Optimizing bulk transfers with software-defined optical wan," in *SIGCOMM*, 2016.

[8] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia: Geo-distributed machine learning approaching lan speeds," in *NSDI*, 2017.

[9] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center tcp (dctcp)," in *SIGCOMM*, 2010.

[10] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang, "Congestion control for large-scale rdma deployments," in *SIGCOMM*, 2015.

[11] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "Tcp vegas: New techniques for congestion detection and avoidance," in *SIGCOMM*, 1994.

[12] R. Mittal, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, D. Zats *et al.*, "Timely: Rtt-based congestion control for the datacenter," in *SIGCOMM*, 2015.

[13] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control," in *ACM Queue*, 2016.

[14] V. Arun and H. Balakrishnan, "Copa: Practical delay-based congestion control for the internet," in *NSDI*, 2018.

[15] A. Production, *https://www.arista.com/en/products*, Accessed in 2019.

[16] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *SIGCOMM*, 2015.

[17] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano *et al.*, "Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network," in *SIGCOMM*, 2015.

[18] S. Bauer, R. Beverly, and A. Berger, "Measuring the state of ecn readiness in servers, clients, and routers," in *IMC*, 2011.

[19] M. Kühlewind, D. P. Wagner, J. M. R. Espinosa, and B. Briscoe, "Using data center tcp (dctcp) in the internet," in *GLOBECOM*, 2014.

[20] netem in Linux Foundation Wiki, *https://wiki.linuxfoundation.org/networking/netem*, Accessed in 2019.

[21] S. Ha, I. Rhee, and L. Xu, "Cubic: a new tcp-friendly high-speed tcp variant," in *SIGOPS*, 2008.

[22] K. Ramakrishnan and R. Jain, "A binary feedback scheme for congestion avoidance in computer networks," in *TOCS*, 1990.

[23] K. Ramakrishnan, S. Floyd, and D. Black, "The addition of explicit congestion notification (ecn) to ip," in *RFC 3168*, 2001.

[24] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," in *SIGCOMM*, 2004.

[25] M. Alizadeh, A. Javanmard, and B. Prabhakar, "Analysis of dctcp: stability, convergence, and fairness," in *SIGMETRICS*, 2011.

[26] J. Corbet, *TSO sizing and the FQ scheduler*, Accessed in 2019, https://lwn.net/Articles/564978/.

[27] K. Nichols and V. Jacobson, "Controlling queue delay," in *ACM Queue*, 2012.

[28] V. Jacobson, "Congestion avoidance and control," in *SIGCOMM*, 1988.

[29] T. V. Lakshman and U. Madhow, "The performance of tcp/ip for networks with high bandwidth-delay products and random loss," in *ToN*, 1997.

[30] P. Brown, "Resource sharing of tcp connections with different round trip times," in *INFOCOM*, 2000.

[31] A. K. Choudhury and E. L. Hahne, "Dynamic queue length thresholds for shared-memory packet switches," in *ToN*, 1998.

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

[33] 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," 1984.

[34] B. Li, K. Tan, L. L. Luo, Y. Peng, R. Luo, N. Xu, Y. Xiong, and P. Cheng, "Clicknp: Highly flexible and high-performance network processing with reconfigurable hardware," in *SIGCOMM*, 2016.

[35] W. Bai, L. Chen, K. Chen, and H. Wu, "Enabling ecn in multi-service multi-queue data centers," in *NSDI*, 2016.

[36] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "pfabric: Minimal near-optimal datacenter transport," in *SIGCOMM*, 2013.

[37] P. Goyal, A. Narayan, F. Cangialosi, D. Raghavan, S. Narayana, M. Alizadeh, and H. Balakrishnan, "Elasticity detection: A building block for internet congestion control," in *arXiv CoRR*, 2018, [Online]. Available: https://arxiv.org/abs/1802.08730.

[38] N. K. Sharma, M. Liu, K. Atreya, and A. Krishnamurthy, "Approximating fair queueing on reconfigurable switches," in *NSDI*, 2018.

[39] B. Cronkite-Ratcliff, A. Bergman, S. Vargaftik, M. Ravi, N. McKeown, I. Abraham, and I. Keslassy, "Virtualized congestion control," in *SIGCOMM*, 2016.

[40] K. He, E. Rozner, K. Agarwal, Y. J. Gu, W. Felter, J. Carter, and A. Akella, "Ac/dc tcp: Virtual congestion control enforcement for datacenter networks," in *SIGCOMM*, 2016.

[41] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," in *SIGCOMM*, 2013.

[42] S. Kopparty, S. V. Krishnamurthy, M. Faloutsos, and S. K. Tripathi, "Split tcp for mobile ad hoc networks," in *GLOBECOM*, 2002.

[43] T. Flach, N. Dukkipati, A. Terzis, B. Raghavan, N. Cardwell, Y. Cheng, A. Jain, S. Hao, E. Katz-Bassett, and R. Govindan, "Reducing web latency: the virtue of gentle aggression," in *SIGCOMM*, 2013.

[44] F. Le, E. Nahum, V. Pappas, M. Touma, and D. Verma, "Experiences deploying a transparent split tcp middlebox and the implications for nfv," in *HotMiddlebox*, 2015.

[45] C. Jin, D. X. Wei, and S. H. Low, "Fast tcp: motivation, architecture, algorithms, performance," in *INFOCOM*, 2004.

[46] K. Tan, J. Song, Q. Zhang, and M. Sridharan, "A compound tcp approach for high-speed and long distance networks," in *INFOCOM*, 2006.

[47] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, and M. Yasuda, "Less is more: trading a little bandwidth for ultra-low latency in the data center," in *NSDI*, 2012.

[48] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-aware datacenter tcp (d2tcp)," in *SIGCOMM*, 2012.

[49] A. Munir, I. A. Qazi, Z. A. Uzmi, A. Mushtaq, S. N. Ismail, M. S. Iqbal, and B. Khan, "Minimizing flow completion times in data centers," in *INFOCOM*, 2013.

[50] S. Han, K. Jang, A. Panda, S. Palkar, D. Han, and S. Ratnasamy, "Softnic: A software nic to augment hardware," in *Technical Report UCB/EECS-2015-155, EECS Department, University of California, Berkeley*, 2015.

[51] C. Lee, C. Park, K. Jang, S. Moon, and D. Han, "Dx: Latency-based congestion control for datacenters," in *ToN*, 2016.

[52] Y. Le, B. Stephens, A. Singhvi, A. Akella, and M. Swift, "Rogue: Rdma over generic unconverged ethernet," in *SoCC*, 2018.

[53] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *SIGCOMM*, 2002.

[54] N. Dukkipati, M. Kobayashi, R. Zhang-Shen, and N. McKeown, "Processor sharing flows in the internet," in *IWQoS*, 2005.

[55] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron, "Better never than late: Meeting deadlines in datacenter networks," in *SIGCOMM*, 2011.

[56] C.-Y. Hong, M. Caesar, and P. Godfrey, "Finishing flows quickly with preemptive scheduling," in *SIGCOMM*, 2012.

[57] D. Han, R. Grandl, A. Akella, and S. Seshan, "Fcp: A flexible transport framework for accommodating diversity," in *SIGCOMM*, 2013.

[58] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal, "Fastpass: A centralized zero-queue datacenter network," in *SIGCOMM*, 2014.

[59] J. Perry, H. Balakrishnan, and D. Shah, "Flowtune: Flowlet control for datacenter networks," in *NSDI*, 2017.

[60] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, “Design, implementation and evaluation of congestion control for multipath tcp,” in *NSDI*, 2011.

[61] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, “Improving datacenter performance and robustness with multipath tcp,” in *SIGCOMM*, 2011.

[62] Y. Lu, G. Chen, B. Li, K. Tan, Y. Xiong, P. Cheng, J. Zhang, E. Chen, and T. Moscibroda, “Multi-path transport for rdma in datacenters,” in *NSDI*, 2018.

[63] I. Cho, K. Jang, and D. Han, “Credit-scheduled delay-bounded congestion control for datacenters,” in *SIGCOMM*, 2017.

[64] M. Handley, C. Raiciu, A. Agache, A. Voinescu, A. Moore, G. Antichi, and M. Wojcik, “Re-architecting datacenter networks and stacks for low latency and high performance,” in *SIGCOMM*, 2017.

[65] B. Montazeri, Y. Li, M. Alizadeh, and J. Ousterhout, “Homa: A receiver-driven low-latency transport protocol using network priorities,” in *SIGCOMM*, 2018.

[66] K. Winstein and H. Balakrishnan, “Tcp ex machina: Computer-generated congestion control,” in *SIGCOMM*, 2013.

[67] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira, “Pcc: Re-architecting congestion control for consistent high performance,” in *NSDI*, 2015.

[68] M. Dong, T. Meng, D. Zarchy, E. Arslan, Y. Gilad, B. Godfrey, and M. Schapira, “Pcc vivace: Online-learning congestion control,” in *NSDI*, 2018.

[69] 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 *NSDI*, 2014.