In-Band Network Telemetry-Based Congestion Control for Cross-Datacenter Networks

Feixue Han, Qing Li, *Senior Member, IEEE*, Rui Huang, Huiling Jiang, and Yong Jiang, *Member, IEEE*

***Abstract*—The cloud has become an indispensable infrastructure for modern enterprises, and cloud-based cross-data center (DC) services are becoming more prevalent. However, most of the existing congestion control (CC) algorithms are designed for individual networks and are ill-suited for cross-DC networks for two reasons. First, intra-DC and cross-DC flows have different control loop lengths. Second, they can hardly share the same parameter settings for buffer-based signals. Concentrating on accurate and rapid CC for cross-DC networks, in this paper, we propose Gear, a novel INT-based CC mechanism that assigns different tasks to intra- and cross-DC flows. With the help of In-Band Network Telemetry (INT), intra-DC flows can determine both their appropriate sending rates and the bandwidth occupation of cross-DC flows, which facilitates their ability to fully utilize the network while keeping low latency. Meanwhile, the cross-DC flows are responsible for ensuring fair bandwidth share between intra-DC and cross-DC flows through proactive bandwidth allocation. Our experimental results demonstrate that Gear significantly optimizes the throughput fairness under cross-DC networks compared with state-of-the-art CC algorithms. Gear also improves throughput by up to 20%.**

***Index Terms*—Data center, transport, congestion control (CC), in-band network telemetry (INT).**

I. Introduction

CLOUDS have become the crucial backbone for modern enterprises. Recent research [1] shows that approximately two-thirds of these enterprises now rely on cloud-based cross-data center (DC) infrastructures (Figure 1), which connect multiple data centers through wide area networks (WAN)

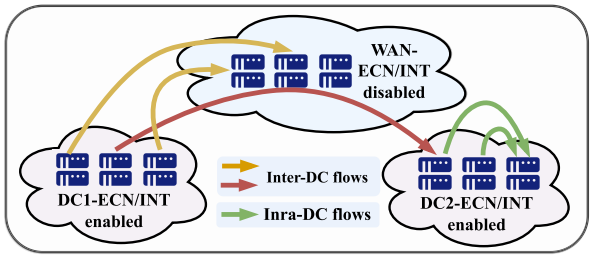


Fig. 1. The structure of cross data center network and the types of flows.

[2], [3]. This geographically distributed network significantly enhances the scalability and resilience of large-scale online applications, such as virtual reality [4], online retailing, and online gaming, while simultaneously reducing their latency. Under such heterogeneous infrastructure, intra-DC flows and cross-DC flows, with vastly distinct characteristics, vie for the bandwidth when the bottleneck is located in the DC.

Ensuring fairness during the interaction between intra-DC and cross-DC flows is of paramount importance since the completion of a request often requires the coordination of these two types of flows. However, CC algorithms have long been designed for individual networks (*i.e.*, all flows run in a DC [5], [6], [7], [8], [9], [10], [11], [12] or WAN network [13], [14], [15], [16], [17], [18], [19]). Taking DCTCP as an example, the ECN threshold setting in the DCTCP algorithm is positively correlated with the bandwidth-delay product (BDP). The RTTs of the links in DC are assumed to be similar, but they demonstrate significant variability in WAN. Switches are unable to observe the RTT of the cross-DC flows and set specific thresholds for them. Meanwhile, a straightforward combination of the CCs of WAN and DC is not a viable approach. If a DCTCP flow and a CUBIC flow vies for the bottleneck bandwidth in DC, the loss-oriented CUBIC flow will show greater competitiveness since the DCTCP always reduces its congestion window (cwnd) before packet loss occurs. We summarize the reasons for the inapplicability of these algorithms as follows.

The fundamental reason is that intra-DC and cross-DC flows have different control loop lengths. When congestion occurs, senders take an RTT to react to the network feedback. Before cross-DC flows respond, intra-DC flows must experience multiple rounds of rate deceleration to alleviate the congestion caused by cross-DC flows. This impairs the throughput and flow completion time (FCT) of the intra-DC flows. Similarly, if all flows increase their cwnd by a packet per RTT, cross-DC flows are at a significant disadvantage when competing with intra-DC flows. A potential solution might involve creating

Received 22 November 2024; revised 10 August 2025; accepted 8 October 2025; approved by IEEE TRANSACTIONS ON NETWORKING Editor X. Luo. This work was supported in part by the Project of the Department of Strategic and Advanced Interdisciplinary Research of Pengcheng Laboratory under Grant 2025QYA001, in part by the National Key Research and Development Program of China under Grant 2022YFB3105000, and in part by the Shenzhen Key Lab of Software Defined Networking under Grant ZDSYS20140509172959989. *(Feixue Han, Qing Li, and Rui Huang contributed equally to this work.) (Corresponding authors: Qing Li; Yong Jiang.)*

Feixue Han is with the Department of Strategic and Advanced Interdisciplinary Research, Peng Cheng Laboratory, Shenzhen 518055, China, and also with Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen 518055, China (e-mail: hanfx19@mails.tsinghua.edu.cn).

Qing Li is with the Department of Strategic and Advanced Interdisciplinary Research, Peng Cheng Laboratory, Shenzhen 518055, China (e-mail: liq@pcl.ac.cn).

Rui Huang and Yong Jiang are with Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen 518055, China, and also with the Department of Strategic and Advanced Interdisciplinary Research, Peng Cheng Laboratory, Shenzhen 518055, China (e-mail: huang-r24@mails.tsinghua.edu.cn; jiangy@sz.tsinghua.edu.cn).

Huiling Jiang is with Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen 518055, China (e-mail: jianghl0826@163.com).

Digital Object Identifier 10.1109/TON.2025.3621518

control loops of equal lengths for both types of flows through forward feedback (like Annulus [20]). Nevertheless, there are certain limitations: (*i*) To guarantee the same loop length, this method necessitates the selection of the feedback node as the bottleneck, which is often variable and not consistent. (*ii*) Forward feedback would compound the burden on the already heavily loaded downlink.

Meanwhile, modern datacenters face fundamental challenges in configuring queue-based congestion signals (*e.g.*, ECN and delay) for cross datacenter networks. Since the optimal ECN or delay threshold should scale with the BDP of a flow [21], any uniform threshold is suboptimal: a low value throttles long RTT cross-DC flows, while a high value inflates queues for short RTT intra-DC flows. Per-flow adaptive thresholds are appealing in theory, yet commodity switches lack per-flow ECN marking, and assigning large delay thresholds to cross-DC flows makes them blind to congestion on shallow-buffer intra-DC paths. Hybrid schemes such as Gemini [22] attempt a compromise, using a shared low ECN threshold while softening window reductions for cross-DC traffic, but this inevitably biases bandwidth toward cross-DC flows and slows convergence.

To address the difficulty of reconciling the performance of intra-DC and cross-DC flows, we believe the key is a good CC algorithm specially designed for cross-DC networks. Our emphasis is on ensuring that both types of flows receive a fair bandwidth share, instead of forcing a consistent rate adjustment frequency. This CC should (*i*) *minimize the impact of different control loop lengths on flow fairness:* The primary reason for unfair bandwidth allocation is the untimely rate adjustment of cross-DC flows. Before cross-DC flows respond to the congestion signal, the rate adjustments that should be executed by cross-DC flows are instead taken on by intra-DC flows. Therefore, beyond the intended adjustments, additional rate adjustments (compensation for intra-DC flows) are necessary for cross-DC flows to promote fairness. (*ii*) *adopt a congestion signal whose parameter configuration is decoupled from the flows' attributes:* we argue that link utilization is a suitable choice, as it characterizes the average network state of a period, rather than instantaneous signals that are susceptible to the impact of flow attributes. The senders can now acquire the link utilization in data centers via In-band network telemetry (INT) technology, which has already become available in new programmable switches [23], [24].

In this paper, we propose **Gear**, a congestion control algorithm for cross-DC networks. Our insight lies in assigning distinct tasks to intra-DC and cross-DC flows. Specifically, intra-DC flows are responsible for driving link utilization to the target and maintaining low latency, while cross-DC flows focus on regulating fairness. With the help of abundant information from INT, intra-DC flows can drive link utilization to the target within two intra-DC RTTs (IRTTs) in scenarios where cross-DC flows fail to respond to link congestion or underutilization in a timely manner (Section III-C). Upon receiving the INT information, cross-DC flows recognize that responding to congestion signals has been delegated to intra-DC flows in the previous cross-DC RTT (CRTT). To ensure fairness, cross-DC flows will double the rate increase or decrease suggested by the congestion signal for a CRTT to compensate for the untimely response (Section III-D). This enables alternate bandwidth allocation between them like a pair of gears and thereby ensuring equitable sharing of network resources.

Our main contributions are summarized as follows:

- Through analysis, we have identified the primary constraints associated with cross-DCN CC algorithms. Specifically, we have highlighted the limitation of relying solely on buffer-based signals to effectively accommodate this scenario.
- We propose Gear, a novel CC algorithm designed specifically for cross-DCN scenarios. Gear utilizes INT information (link utilization) as the congestion signal to enable alternate bandwidth allocation between intra- and cross-DC flows.
- We implement Gear in the ns-3 simulator and build a prototype consisting of servers and P4 switches. We conduct comprehensive experiments to show that Gear outperforms other algorithms in terms of fairness, throughput, and latency, including HPCC [7], DCTCP [5], GEMINI [22] and Timely [9].

Our experimental results show that Gear significantly improves the fairness between intra- and cross-DC traffic. Besides, Gear can reduce the duration of Priority Flow Control (PFC) by up to 5x, while achieving a 20% higher throughput.

II. Context

In this section, we highlight the heterogeneity between WAN and DC networks in II-A and show that buffer-based signals can hardly apply to cross-DC networks (cross-DCN) in II-B.

A. Heterogeneity Between Wan and DC Networks

A cross-DCN is comprised of numerous geographically dispersed DC networks that are interconnected via a WAN. Whereas DC and WAN networks exhibit strong heterogeneity, primarily manifested in the following three aspects:

Control loop lengths. The majority of current CC algorithms adjust the sending rate at intervals of one RTT. Thus, the control loop length is closely tied to the RTT. In DC networks, RTTs are typically in the range of 10–50us, while in WANs, RTTs can span from 10ms to over 100ms depending on distance and routing policies, as observed in GEMINI [22]. Consequently, there is a disparity of 3–4 orders of magnitude in control loop length. This results in a disparity of three to four orders of magnitude in control loop granularity. Such extreme differences give rise to RTT unfairness: intra-DC flows can respond to the congestion signal hundreds to thousands of times before cross-DC flows when they are bottlenecked together; similarly, intra-DC flows can seize more bandwidth within the same timeframe, potentially leading to cross-DC flow starvation.

Buffer size requirements. To fully utilize the link, flows require buffers that are proportional to their BDP [21] (the advancement of the CC algorithm can optimize the required buffer size while still conforming to this pattern [25]). Therefore, even with the same rate, the buffer size required for cross-DC flows is several hundred to thousand times larger than that of DC flows.

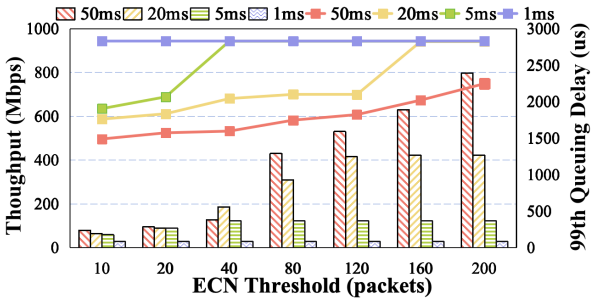


Fig. 2. Throughput and queuing delay of flows with different base RTTs under networks with varying ECN marking thresholds. Queuing delay is the difference between RTT and base RTT.

Furthermore, flows within DC have strict latency requirements, as evidenced by [26], hence, DCs tend to avoid excessively long queues. A report [27] reveals that WAN switches typically employ deep buffers (tens of megabytes per port per Gbps), whereas data center routers are equipped with shallow-buffer switches (tens of kilobytes per port per Gbps).

Traffic characteristics. Since DC hosts plenty of distributed applications, *e.g.*, Spark [28] and Hadoop [29], traffic in DC demonstrates characteristics of scatter and gather [30]. The traffic is partitioned into small chunks, each processed by different servers, and the resulting outputs are later aggregated. Under this pattern, DC traffic exhibits bursty characteristics and generates severe fluctuations over sub-millisecond timescales. Nevertheless, cross-DC flows that experience millisecond-level RTTs can hardly track the real-time available bandwidth.

B. Buffer-Based Signals and Algorithms in Cross-DCN

Common buffer-based signals: The currently commonly used buffer-based signals include ECN [5], [6] and delay [9], [10], [14], which reflect the instantaneous queue length at a time. Threshold selection is a critical factor in achieving optimal flow performance when utilizing signals for congestion control. Typically, the thresholds for ECN and delay are closely related to the characteristics of the flows, often proportional to their respective BDP or delay. In this section, we conducted an ns3-simulation experiment on a 1Gbps link to investigate the impact of ECN thresholds on the throughput and queuing delay of flows with varying base RTTs. Figure 2 displays the experimental results, where the line charts correspond to the left axis for throughput, and the bar charts correspond to the right axis for the 99th percentile queuing delay. Flows with larger base RTTs require larger ECN thresholds to fully utilize the link. However, the growth in throughput is not strictly proportional to the ECN threshold. The 99th percentile queuing delay exhibits a noticeable increase as the ECN threshold is raised, even when the flow does not fully occupy the bandwidth, as evidenced by the red line and bar. This indicates that an increase in ECN threshold exacerbates network jitter.

In data centers, the base RTTs of all paths are generally uniform, allowing the sharing of a common threshold among all flows. However, in cross-DC networks, identifying an appropriate and universal threshold for buffer-based signals presents a formidable challenge. Taking ECN marking as an example, a low-level ECN marking that is appropriate for DC flows may lead to throughput degradation for cross-DC flows, while a high-level ECN marking that can ensure high throughput for cross-DC flows may result in intolerable latency for DC flows. Despite setting different thresholds for different flows seems feasible, flows with different thresholds are queued together, which means flows with larger thresholds will negatively impact the queuing delay of the flows with smaller thresholds.

Buffer-based cross-DC algorithms: Practitioners have also put forth CC algorithms [20], [22], [31] tailored for the cross-DCN scenarios. GTCP is controlled by the receiving end and the sending end when congestion occurs in the data center and WAN, respectively. We argue that the two-end algorithm introduces excessive modifications to the protocol stack, thus primarily focusing on the sender-driven algorithm: Genmini and GTCP. These algorithms employ two distinct signals to pinpoint the location of congestion: the ECN signal for the DC network and the delay for the WAN network. Despite their efficiency in locating the bottleneck, the combined usage of these signals is still inadequate in addressing the challenge of parameter configuration that necessitates coupling with the attributes of the flows.

GEMINI aims to maintain a high throughput of cross-DC flows in shallow buffer networks, which proportionally links the additive increase (AI) factor of the congestion window with the RTT. This theoretically allows cross-DC flows to compete competitively with intra-DC flows. However, there are still issues with this algorithm. 1) Expecting to use an aggressive AI factor, GEMINI also reduces the penalty of ECN marking for cross-DC flows. This results in a bias towards cross-DC flows in terms of throughput. 2) Due to the potential for RTT to differ by hundreds or even thousands of times, cross-DC flows may have an extremely large AI factor, which can cause severe rate and delay fluctuations. To address this, GEMINI imposes an upper limit on the AI factor at the expense of sacrificing theoretical fairness. During the replication process of GEMINI, we spent a considerable amount of time tuning it to make its performance acceptable. We present the performance of GEMINI under different upper limit restrictions in the Appendix VII.

Annulus creates a near-source control loop for all flows, allowing flows to receive congestion signals with the same delay. This ensures that intra-DC flows do not excessively reduce their rates due to their fast response rate. However, Annulus ignores that cross-DC flows also have lower rate growth frequencies compared to intra-DC flows. Even taking into account that all flows initiate with line rate, this issue will still be exposed due to the dynamic nature of the network. Therefore, simply replicating the content of the paper [20] makes it difficult to achieve fair bandwidth allocation between cross- and intra-DC flows. Additionally, Annulus assumes that congestion always occurs at the near-source (ToR) switches, thus lacking the capability to respond to congestion located at the aggregation or core layer.

Based on these limitations, we propose an algorithm based on INT that avoids the need to set different parameters for

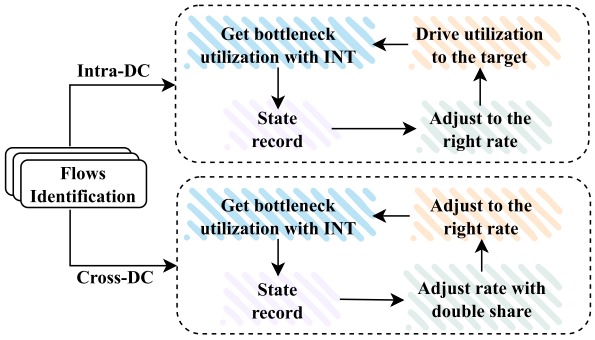


Fig. 3. The structure of Gear. The intra-DC flows record the current state and adjust to the right rate in the first IRTT, and then they drive the bottleneck utilization to the target in the second IRTT. After receiving the INT information, the cross-DC flows record the current state and make a rate adjustment with a double share in the first CRTT. Then it restores the sending rate to the right rate in the second CRTT.

different types of traffic and achieves fair bandwidth allocation without the need for multiple iterations in the absence of traffic isolation.

III. Design

A. Overview of Gear

Our insight is to enable alternate bandwidth allocation between cross-DC flows and intra-DC flows, synchronized with the CRTT, thereby ensuring equitable bandwidth sharing of network resources. Gear assigns distinct tasks to intra- and cross-DC flows: intra-DC flows are responsible for the low latency and high throughput, while cross-DC flows are responsible for fairness.

The brief workflow of Gear is shown in Figure 3. Gear adopts INT as the congestion signal in data centers. Upon the initiation of a new flow, Gear first determines whether it is an intra-DC or cross-DC flow in order to execute the corresponding logic: 1) *Logic of the intra-DC flow* (Section III-C): Upon receiving a packet with INT information, Gear calculates the utilization of the bottleneck port along the path, records the current sending rate and utilization, and adjusts the rate accordingly. However, due to the delayed response of cross-DC flows, a single rate adjustment by intra-DC flows is insufficient for the bottleneck to achieve the target utilization. Therefore, intra-DC flows monitor the change in bottleneck utilization resulting from the current rate adjustment after an RTT. Based on this change, they calculate the aggregated rates of cross-DC flows and adjust their sending rates again to drive the utilization to the target. 2) *Logic of the cross-DC flow* (Section III-D): After a CRTT, cross-DC flows receive the corresponding INT information and calculate the share by which the rate should increase or decrease. To compensate for the delayed response, cross-DC flows record the current sending rate and the share, and then adjust the rates according to $rate \pm 2 \times share$. In the next CRTT, the flows restore the rates to $rate \pm share$.

B. Identifying Flows and Their Bottlenecks

Flow identification: The most significant distinction between the intra-DC and cross-DC flows is their substantial RTT difference that can range widely across several orders of magnitude. Such a difference can be exploited to segregate cross-DC flows from intra-DC flows by the usage of an RTT threshold. The RTT, measured during the three-way handshake phase, is compared against the threshold ($k = 100$ by default) to determine if it is an intra-DC flow or a cross-DC one. Since the handshake packets can suffer from severe congestion, leading to intra-DC flow misclassification as cross-DC flows, a corrective approach is employed by checking if the subsequent measured RTT values fall below the threshold. Note that the efficiency of the algorithm is preserved even if a low-latency cross-DC flow is incorrectly labeled.

Determine the bottleneck-hop with INT: The INT information brought by acknowledgments (ACKs) includes timestamp (ts), queue length (ql), transmitted bytes (tx), and link bandwidth capacity (bw). The availability of this information allows us to compute the link utilizations of each switch along the path and identify the highest utilization among them, which designates the bottleneck hop. Suppose p is a data packet, and p_ack is its corresponding ACK. The INT information of the $i-th$ link received when p is transmitted and carried by p_ack are denoted as $p.L_i$ and $p\_ack.L_i$, respectively. The link utilization u_i can be calculated using the following:

$$rate_i = \frac{p\_ack.L_i.tx - p.L_i.tx}{p\_ack.L_i.ts - p.L_i.ts}$$
$$u_i = \frac{rate_i}{p\_ack.L_i.bw} + \frac{p\_ack.L_i.ql}{baseRtt \cdot p\_ack.L_i.bw} \quad (1)$$

Then the link utilization of the bottleneck is:

$$u_{btl} = \max u_i \quad (2)$$

Compared with queue-based signals which are susceptible to instantaneous queue length, INT offers an RTT-agnostic alternative. To prevent packet drops during transmission over the Internet, INT entries are filled in the TCP option field. Nevertheless, the 40-byte capacity constraint of the TCP option field restricts the number of INT entries that can be accommodated. As a result, when the number of traversed switches surpasses the maximum number of INT entries, overwriting one of the previous INT entries becomes indispensable. Intuitively, we should overwrite the INT information of low-utilization links with that of high-utilization links. However, performing such calculations is challenging for programmable switches due to their inability to maintain extensive flow states or conduct intricate calculations. To address these challenges, we adopt a two-pronged approach. Firstly, we endeavor to preserve as many INT entries as possible. Secondly, we overwrite the information based on the current state. We directly append a new INT entry when space remains available in the option, and when space is unavailable, we compare the time required for consuming the current queue for two switches, maintaining the entry with the longer time requirement as follows:

$$L_i = \arg\max_{L_i} \left\{ \frac{L_i.ql}{L_i.bw} \right\} \quad (3)$$

Note that due to the lack of division support in P4, we substitute the division with shift operation in the actual calculation process.

C. CC for Intra-DC Flows

The task of intra-DC flows is to swiftly adapt the link utilization to the target. This adaption encompasses three steps: *I.* All the intra-flows adjust their sending rate to the right rate[1] based on the INT information. *II.* Inferring the proportion of cross-DC traffic based on the network feedback of step I. *III.* Intra-flows readjust their sending rate to drive the link utilization to the target utilization.

Step I: With the assistance of the fine granularity INT information, a flow can move to the right rate with just one rate update, *i.e.*, dividing the current sending rate by the bottleneck link utilization. Since each ACK carries INT information, reacting to them all will cause the issue of overreaction. Referenced by HPCC, we maintain two congestion windows: a reference window w^c and a current window at time t $w(t)$. w^c is updated to $w(t)$ only after an RTT has passed since the last update, and $w(t)$ is always updated in a per-ACK manner based on w^c. Gear transmits data according to the size of $w(t)$.

We use $u(t)$ to denote the link utilization calculated from the INT information carried by the ACK received at time t. Assuming a target bandwidth utilization of η, Gear adjusts the sending rate as follows:

$$w(t) = w^c \cdot \frac{\eta}{u(t)} + W_{AI} \tag{4}$$

where W_{AI} is a additive factor. We denote the time interval between two consecutive multiplicative increase/decrease (MI/MD) operations of a flow as T. As shown in line 2 in Algorithm 1, when a flow encounters congestion ($T > IRTT$) or has performed additive increases for $maxStage$ times ($T = maxStage \cdot IRTT$), it performs a multiplicative operation. Otherwise, the sender performs additive increases with the fairness factor w_{AI} to avoid oversensitive and promote fairness. It should be noted that the innovation of Gear does not lie in how to calculate the right rate based on INT, but in leveraging INT to optimize performance in scenarios where cross-DC and intra-DC flows coexist (including fairness, link utilization, etc).

Step II: Since cross-DC flows will keep their reference window unchanged for a CRTT (much longer than IRTT), just one adjustment of the DC flow rates can not push the link utilization to the target. If $u(t) > \eta$, both the cross-DC and intra-DC flows experience the extra queuing latency by the cross-DC flows; while if $u(t) < \eta$, the bandwidth that can not be occupied in time would result in the continuous link underutilization, which shows that intra-DC flows require supplementary modifications to attain optimal link utilization and minimal latency. Therefore, we try to determine the aggregated bandwidth share for cross-DC flows, allowing intra-DC flows to take on the bandwidth share that should be increased/decreased by the cross-DC flows.

Simply put, Gear calculates the bandwidth share of cross-DC flows by monitoring the impact of the intra-DC flow rate adjustments on network utilization. We use w_a^c, and w_b^c to represent the aggregated reference window of intra-DC and cross-DC flows, respectively. Suppose all intra-DC flows simultaneously adjust their windows, we have:

$$\begin{aligned} w_a^c + w_b^c &= u(t) \cdot bw \\ \frac{\eta}{u(t)} w_a^c + w_b^c &= u(t + IRTT) \cdot bw \end{aligned} \tag{5}$$

In Section V-B, we analyze the case where flows make adjustments asynchronously. This can improve the estimation accuracy of w_a^c and w_b^c while also increasing the computational overhead.

Solving Equation 5, we have:

$$w_b^c = u(t) \cdot \frac{u(t + IRTT) - \eta}{u(t) - \eta} \cdot bw \tag{6}$$

Considering that intra-DC flows may update their rates at any time during (*t*, *t* + IRTT), we process the utilization within

Algorithm 1 Sender Algorithm for an Intra-DC Flow a_1 and a Cross-DC Flow b_1

1 **function** `IntraCmptWind(`*u(t), updatew*^{c}`)`
2 **if** $u(t) \geq \eta$ ***or*** $incStage >= maxStage$ **then**
3 **if** *flag* **then**
4 $w_{a_1}(t) = \frac{\eta}{u(t)} w_{a_1}^c + w_{AI}$
5 $t_{prev} = t$
6 **else**
7 compute w_a^c, w_b^c with $u(t)$ and $u(t_{prev})$
8 $w_{a_1}(t) = Eq.\ 7 + w_{AI}$
9 **if** $updatew^c$ **then**
10 $incStage = 0;\ w_{a_1}^c = w(t)$
11 $flag = !flag$
12 **else**
13 $w(t) = w_{a_1}^c + w_{AI}$
14 **if** $updatew^c$ **then**
15 $incStage++;\ w_{a_1}^c = w(t)$
16 **return** $w_{a_1}(t)$
17 **function** `CrossCmptWind(`*u(t), updatew*^{c}`)`
18 $w_{b_1}(t) = (\frac{2\eta}{u(t)} - 1) w_{b_1}^c + \frac{CRTT}{IRTT} W_{AI}$
19 **if** $updatew^c$ **then**
20 $u(t) = Eq.\ 10$
21 $w_{b_1}^c = \frac{\eta}{u(t)} w_{b_1}^c$
22 **return** $w_{b_1}(t)$
23 **procedure** `NewACK(`*ack*`)`
24 **if** $ack.seq > lastUpdateSeq$ **then**
25 $updatew^c = true$
26 $lastUpdateSeq = snd\_nxt$
27 **else**
28 $updatew^c = false$
29 **if** $rtt < k \times basertt$ **then**
30 $w(t) = IntraCmptWind(u_{btl}(t), updatew^c)$
31 **else**
32 $w(t) = CrossCmptWind(u_{btl}(t), updatew^c)$
33 $w(t) = min(w(t), w_{wan}(t))$

[1]The right rate refers to the fair bandwidth share of the bottleneck link.

the time range (t + IRTT, t + 1.5IRTT) in an exponential weighted moving average (ewma) manner, and use this value as the final value of $u(t + IRTT)$ in Equation 5 and 6.

Step III: Now we already have the estimation of w_b^c and can make a secondary adjustment to intra-DC flows to achieve the target link utilization in the next IRTT. If we use a_1 to represent a intra-DC flow, the new reference window $w_{a_1}^c$ for the flow can be expressed as:

$$w_{a_1}^c = \frac{\eta}{u(t)} w_{a_1}^c + \left(\frac{\eta}{u(t)} - 1\right) \frac{w_b^c \cdot w_{a_1}^c}{w_a^c} \tag{7}$$

The whole rate adjustment process of Gear is shown in Algorithm 1. Line 23 to 33 in Algorithm 1 illustrate the process of each newly arrived ACK. Each newly arrived ACK triggers the NewACK procedure. If the sequence number in the incoming ACK is larger than the lastUpdateSeq, we trigger a new synchronization between w^c and $w(t)$. Line 1 to 16 shows the CC at the sender side of an intra-DC flow. We use a flag to indicate whether the rate adjustment considers cross-DC traffic (lines 6-8, according to Equation 7) or not (lines 3-5, according to Equation 4).

If the link is underutilized, the intra-DC flows would temporarily occupy the bandwidth that belongs to the cross-DC flows for a time of $(CRTT - IRTT)$. If congestion occurs, the intra-DC flows take on all the rate reduction tasks to relieve congestion. It is worth noting that temporarily giving up bandwidth for cross-DC flows does not significantly impact the throughput of intra-DC flows. According to Facebook's report [32], the ratio of intra-DC to cross-DC traffic is approximately 5:1. Therefore, for instance, to reduce the link utilization from 110% to 100%, DC flows only need to temporarily yield an additional 1.67% of bandwidth. Google datacenters [33] report an even higher ratio of 9:1, in which case intra-DC flows would only need to yield 1.11% of bandwidth. These adjustments represent a minimal reduction in intra-DC bandwidth share and do not meaningfully degrade FCT for intra-DC flows.

D. CC for Cross-DC Flows

During a CRTT period, cross-DC flow b_1 maintains an unchanged $w_{b_1}^c$ and only makes minor adjustments on $w_{b1}(t)$ according to $w_{b1}(t) = \frac{\eta}{u(t)} \cdot w_{b1}^c$. The rate variation can be expressed as:

$$\Delta = \frac{\eta}{u(t)} w_{b1}^c - w_{b1}^c \tag{8}$$

However, during this period, intra-DC flows undertook the bandwidth share that cross-DC flows should ramp up/down. For example, when there is congestion, intra-DC flows quickly adjust their rates to push the link utilization to the target, which means intra-DC flows transmitted $\Delta \cdot (CRTT - T)$ fewer bytes than they should have sent. Conversely, if there is spare bandwidth, intra-DC flows send $\Delta \cdot (CRTT - T)$ more bytes. Therefore, except for moving to the correct rate, flow b_1 should compensate for the unfinished task in the previous CRTT, *i.e.*, the rate change should be 2Δ:

$$w_{b_1}(t) = w_{b_1}^c + 2\Delta = \left(\frac{2\eta}{u(t)} - 1\right) w_{b_1}^c \tag{9}$$

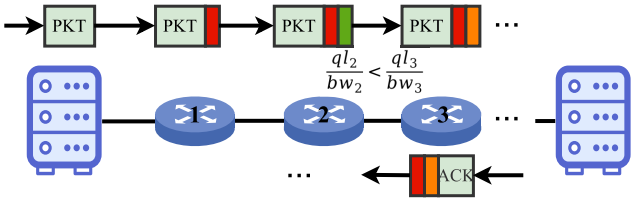


Fig. 4. The addition and replacement of the INT information. When there is insufficient space in the TCP Option, we retain the INT information of the switch with the longest queuing delay.

Note that this formula is applicable to both rate increases and decreases. When the bottleneck link is congested, Δ is a positive value, whereas when there is spare bandwidth, Δ is a negative value.

At the start of each CRTT, we need to update the $w_{b_1}^c$. However, it is important to note that the temporary bandwidth fluctuations resulting from doubling the rate variation can affect the link utilization $u(t)$ (before intra-DC flows adapt to the available bandwidth). Therefore, we proactively mitigate the impact of this additional rate variation on $u(t)$ as follows:

$$\begin{aligned} \Delta u(t) &= \frac{\Delta}{bw} \\ u(t) &= u(t) - \Delta u(t) \end{aligned} \tag{10}$$

Then, $w_{b_1}^c$ is assigned with the correct rate $\frac{\eta}{u(t)} \cdot w_{b_1}^c$, rather than the current sending rate $w(t)$. This ensures the cross-DC flow will back to the correct rate after compensating for the slow response.

Lines 17 to 22 show the CC at the sender side of the cross-DC flows. Since cross-DC flow passes both the data center and WAN network, cross-DC flows have to distinguish whether the bottleneck is located in the data center or WAN. If there are multiple bottlenecks, it reacts to the most congested one (line 33) to satisfy the maximum minimum fairness allocation [34] between the intra-DC and cross-DC flows. When congestion occurs in the WAN network, cross-DC flows adopt the traditional additive increase multiplicative decrease (AIMD) algorithms like CUBIC [13], which does not require any INT or specialized support from WAN switches. When the congestion is located in the data center, cross-DC flows adjust their rates in a multiplicative increase multiplicative decrease (MIMD) manner (line 18) to adapt to the rapid changes in the data center network. After compensating for the slow response, it restores to the right rate (line 21).

IV. Implementation and Evaluation

In this section, we evaluate the performance of Gear through large-scale ns3 simulations and small-scale prototype-based experiments.

A. General Implementation

TCP Option in Gear. To support Gear, we design a new TCP Option, as illustrated in Figure 5. We set the *kind* variable in the TCP Option, which is used to indicate the category of TCP Options, to 253 as it is an undefined value reserved for the experimental purpose. *Len* variable indicates the total length of the Option field, except the padding bytes. For each P4

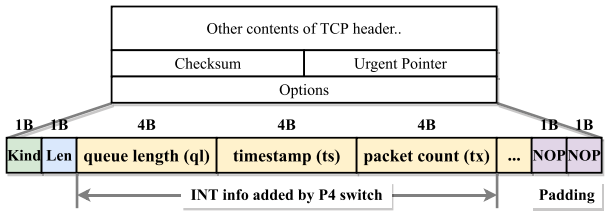


Fig. 5. The TCP Option field of Gear.

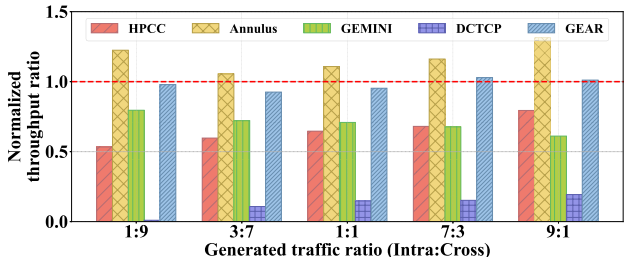


Fig. 6. The x-axis shows the ratio of the traffic sizes of the cross-DC and intra-DC flows generated by the senders. The y-axis shows the normalized ratio of the aggregated throughput of the intra-DC and cross-DC flows. The red dotted line represents the ideal ratio.

switch, its queue length, the egress timestamp, and the number of packets processed are contained in *ql*, *ts*, and *tx* variables, respectively, each with a size of 4 bytes. As the length of the TCP Option needs to be multiple of 4, we add two NOP bytes at the end of the TCP Option for alignment purposes. The TCP Option can be up to 40 bytes long. We believe the port bandwidth of the switches in the DC remains constant. Therefore, the bandwidth parameters can be pre-stored in a configuration file and not included in INT.

Contrast Algorithms. In this paper, we conduct a comparative analysis of Gear against several other CC protocols, namely DCTCP [5], GEMINI [22], Annulus [20], and HPCC [7]. Due to the limited support for end-to-end ECN marking and INT information in WAN environments, ECN and INT are *only enabled on intra-DC switches*. Note that all the receivers are able to mirror the ECN marking or INT information. DCTCP and GEMINI leverage ECN to convey congestion information to end-hosts and adjust the cwnd based on the fraction of marked packets. As GEMINI and Annulus are not open source, we reproduce these algorithms based on their research papers [20], [22]. HPCC employs more precise INT information to decrease the iteration count.

The parameter settings of these algorithms are described as follows: (*i*) As DCTCP suggested, the buffer occupancy threshold for setting the CE-bit is 900KB (600 packets) for 100Gbps links. Considering that if cross-DC and intra-DC flows use the same AI factor, cross-DC flows can hardly obtain any bandwidth, *we set the ratio of the AI factor between cross-DC and intra-DC flows to min(500, CRTT/IRTT).* (*ii*) The target link utilization of INT-equipped algorithms is 95%. (*iii*) For intra-DC traffic, we set the retransmission timeout (RTO) value as 1ms. For cross-DC traffic, we adopt an RTO of 200ms. If not explicitly stated, the bottleneck bandwidth is uniformly set to 100Gbps. (*iv*) For GEMINI, the parameter β for window reduction is set to 0.2, while H for window growth is set to 3e-7 to lead to $h = 1$ (100Gbps link, the measured minimum delay is about 32us). (*v*) For Annulus, the parameter w in $F_b = (Q - Q_{eq}) + w \cdot (Q - Q_{old})$ is set to 8. The sampling interval on TOR switches is set to 15KB. Following the recommendation, we set GD to 1/128. Since Annulus did not specify its CC algorithm at the sender, we applied the DCQCN algorithm for congestion in the data center and the Vegas algorithm for congestion in WAN. The ratio of AI factor is the same as DCTCP.

B. Large-Scale Ns3 Simulations

Setup. We use the fat-tree [35] topology to conduct our simulation. The fabric interconnects 128 servers organized into 4 pods. Each pod consists of four aggregation switches and four top-of-rack (ToR) switches. Aggregation switches are connected to 8 core switches, resulting in a fabric with an overall 4:1 oversubscription. The fat-tree network is directly connected to the cross-DC network with the border router (core routers). We use 100Gbps point-to-point Ethernet links across our entire network. All the switches in the topology have a per-port buffer capacity of 10MB. The base propagation delay between two intra-DC servers on different pods is 20us. To incorporate the effect of long RTT of cross-DC traffic, we set the propagation delay of links connecting to external switches to 25ms, resulting in an RTT of roughly 50ms for cross-DC traffic. We use equal-cost multi-path routing (ECMP) as the default routing strategy.

Workload. We simulate empirical workloads derived from the distribution observed in production data centers. Specifically, we consider two flow size distributions: WebSearch [6] and FB_Hadoop [32], which are widely recognized and publicly available. In most of our experiments, we generate a combination of intra-DC and cross-DC traffic in a proportion of approximately 5:1, as referenced in Facebook's production network. We also adjust the traffic ratio of two types of flows to evaluate the fairness of Gear. We adjust the flow generation rates to set the average link loads to 30%. All the source-destination pairs are generated randomly and uniformly across all of the hosts.

1) Throughput Under Different Workloads: We perform an experiment to observe the bandwidth allocation between intra- and cross-DC flows when they compete for the bottleneck bandwidth inside the DCN. We alter the generated traffic ratio of intra- and cross-DC and observe the average throughput of the two types of flows. Ideally, the average throughput of intra-DC and cross-DC should be consistent with the generated traffic ratio. Therefore, we measure the performance of the five algorithms with the following metric:

$$\text{Normalized throughput ratio} = \frac{\text{Intra\_thpt/Cross\_thpt}}{\text{Intra\_gen/Cross\_gen}}$$

where the Intra_thpt and Cross_thpt represent the counted throughput of intra-DC and cross-DC traffic. Intra_gen and Cross_gen represent the corresponding generated traffic. The flow sizes are generated based on the Web Search distribution, with intra-DC and cross-DC traffic generated in the ratios of [1:9, 3:7, 1:1, 7:3, 9:1]. We measure the throughput of each flow in units of 2RTTs. The experimental results are shown in the Figure 6. The algorithm achieves better fairness if the normalized throughput ratio is closer to 1 (the red

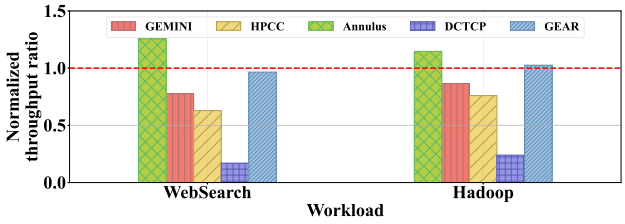
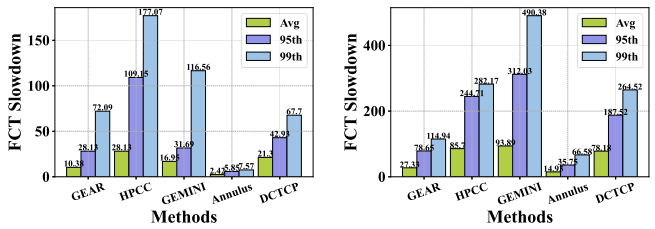


Fig. 7. The ratio of the generated intra- and cross-DC traffic size is 5:1 [20]. The flow sizes are generated according to the workload on the x-axis. The y-axis shows the normalized ratio of the aggregated throughput of flows.



(a) Small flows FCT slowdown (b) Large flows FCT slowdown

Fig. 8. FCT slowdown of various methods under small and large flows.

TABLE I

THE DURATION OF PFC PAUSES UNDER THE FIVE ALGORITHMS. THE RESULTS ARE NORMALIZED TO GEAR

	Gear	HPCC	DCTCP	GEMINI	Annulus
Normalized PFC duration	1	1.87	3.22	5.82	0

dotted line). Except in scenarios where the traffic ratio is extremely unfair (1:9 or 9:1), Gear achieves relatively fair performance. The ratios of GEMINI, HPCC, and DCTCP are all less than 1, which means intra-DC flows are less competitive than cross-DC flows. In HPCC, this can be attributed to the fact that intra-DC flows rapidly engage in multiple rounds of MD operations until the cross-DC flows react. In DCTCP, the throughput ratio increases as the generated traffic ratio increases, but cross-DC flows always grab most of the bandwidth with the adjusted AI factor. GEMINI allows cross-DC flows to have a faster growth rate and a lower rate of decrease, giving them an advantage. Conversely, the ratios of Annulus are greater than 1 because cross-DC flows experiment more frequent rate decrease. Gear shows the best fairness performance across all the generated traffic ratios since Gear is always closest to the red line. This is due to Gear's effective usage of the INT information to accelerate convergence while maintaining fairness.

2) Throughput Under Different Traffic Distribution: As shown in Figure 7, the performance trends of different algorithms are similar to those in Figure 6. However, in the case of Hadoop workload, the unfairness between the two types of flows is alleviated. The reason is that a larger proportion of intra-DC flows are small flows that start at a line rate and tend to complete in a single RTT (before they can experience the impact of unfairness). In both workloads, Gear remains close to the ideal red dotted line, demonstrating stable and consistent fairness performance that is only slightly affected by the workload.

3) The Fraction of the PFC Duration: Priority Flow Control (PFC) will trigger pauses on all upstream interfaces once it detects the possibility of packet loss. Since PFC can spread to multiple hops, it has the potential to cause detrimental effects on networks. We record the total duration of PFC pauses under web search workload and a total load of 30%. As Table I shows, Gear presents the minimal PFC duration and the results of other algorithms are normalized to Gear. During the experiment, the main source of PFC activations is observed to be the line-rate initiated cross-DC flows. DCTCP, GEMINI, and Annulus decrease their rates only after the traffic has jammed, which triggers more PFC pause frames. Note that the near-source feedback method in Annulus allows for faster response to congestion, eliminating the triggering of PFC. HPCC leverages INT to make adjustments before congestion forms (once the utilization reaches 95%), but the latency adjustment of the cross-DC flow rates still enhances the PFC risks. Gear's intra-DC flow considers the cross-DC flows when adjusting their sending rates, enabling them to quickly push the link utilization to the target. This significantly reduces the occurrence of PFC.

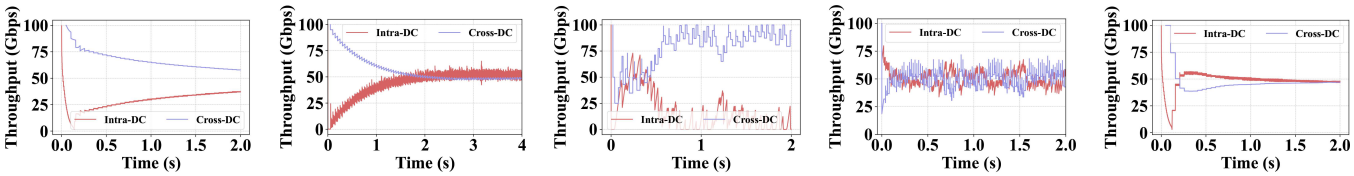
4) FCT Slowdown: Due to the low proportion of large flows in the Hadoop workload, we record the FCT slowdown (the ratio of FCT and the ideal FCT) under the WebSearch workload. We count the average, 95th, and 99th percentile FCT slowdown for small and large flows within the data center for observation purposes. We classified the bottom 50% of flow sizes as small flows and the top 50% as large flows. Figure 8(a) and Figure 8(b) present the performance of the tested methods for small flows and large flows, respectively.

DCTCP, GEMINI, and HPCC are more biased toward cross-DC traffic during their life cycle. Therefore, they show a significant FCT slowdown, especially for large flows. In contrast, cross-DC flows in Annulus exhibit slower growth but can quickly respond to congestion, showing the lowest FCT slowdown. Since all flows are initiated at the line rate, GEMINI's bias toward cross-DC flows requires several adjustment rounds before this becomes apparent. Therefore, GEMINI shows a more pronounced increase in FCT slowdown for large flows.

C. Micro-Benchmark Experiments

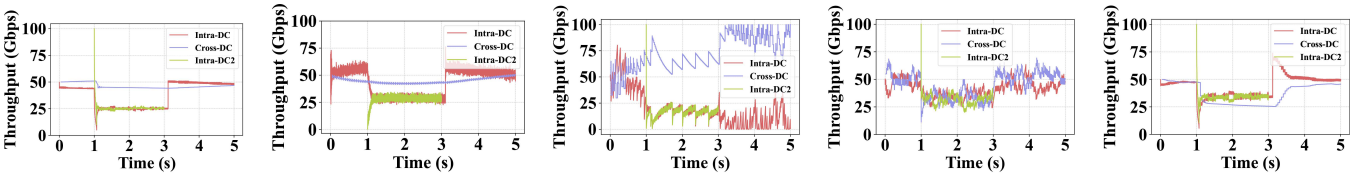
1) Co-Existence of Intra-DC and Cross-DC Flows: Current INT-based algorithms perform well in individual data center networks. Hence, in this section, we mainly focus on the throughput and latency when the two types of flows co-exist in a cross-DCN. We build a dumbbell topology and run two flows: an intra-DC flow and a cross-DC flow that share the same bottleneck (100Gbps). Then we observe the change in the latency and throughput when a new flow is added or ended midway.

As shown in Figure 9, we initiate an intra-DC and a cross-DC flow at 0s. Both flows start at the line rate and last for 2s (GEMINI runs for about 4s because it just reaches convergence at around 2s). We annotated the ratio of throughput between the intra-DC flow and the cross-DC flow in the titles of each subfigure in Figure 9. The results indicate that except for Gear and Annulus, all other algorithms exhibited a certain bandwidth imbalance. We can observe that Gear is the fastest in alleviating congestion, with only a brief rate spike



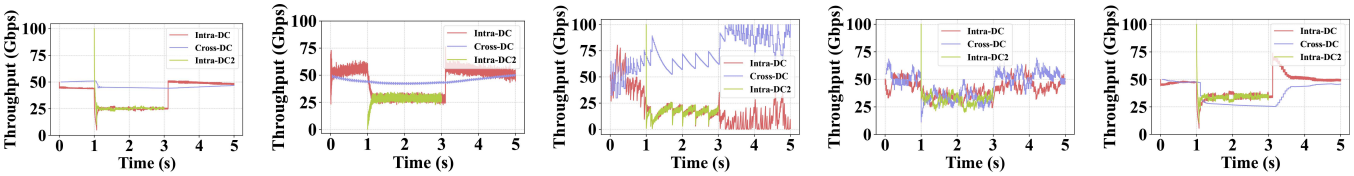
(a) HPCC-0.43 (b) GEMINI-0.769 (c) DCTCP-0.22 (d) Annulus-1.18 (e) Gear-1.02

Fig. 9. The throughput of an intra-DC flow and a cross-DC flow when they compete for the same bottleneck. The bottleneck link capacity is 100Gbps. The throughput ratio of intra-DC to cross-DC flow are marked in the titles of the subfigures.



(a) HPCC (b) GEMINI (c) DCTCP (d) Annulus (e) Gear

Fig. 10. The variation of the throughput when a new flow joins or leaves. An intra-DC flow and a cross-DC flow start at 0s, then another intra-DC flow (line rate) joins at 1s and leaves at 3s.



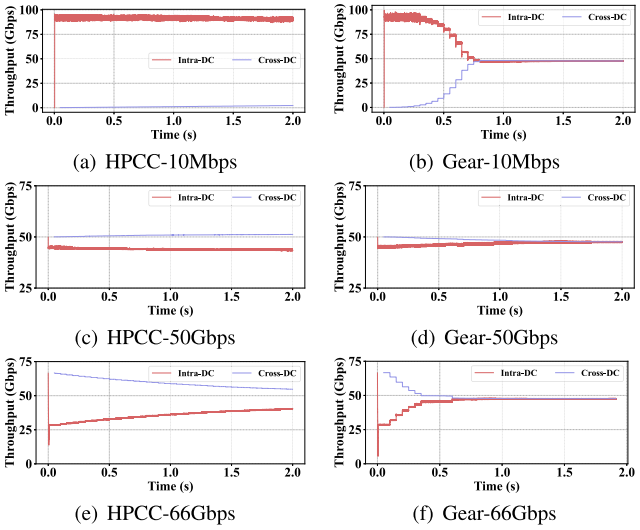
(a) HPCC (b) GEMINI (c) DCTCP (d) Annulus (e) Gear

Fig. 11. The variation of the throughputs when there are only intra-DC flows. A new intra-DC flow is added every second during the time interval of 0-3 seconds, while an intra-DC flow departs every second during the time interval of 4-6 seconds.

during flow line-rate initiation. To achieve fairness, Annulus requires different AI factors for flows with varying RTTs, as described in Section IV-A. In HPCC, the intra-DC flows underwent multiple rounds of slowdown before the cross-DC flows responded, resulting in slow convergence starting from significantly different positions. HPCC flows did not converge to a fair point till the flows ended (at 2s) and caused significant bandwidth loss for the intra-DC flow. Despite that GEMINI achieves fair convergence around 2s, the intra-flow experiences severe throughput impairment before the convergence, leading to a significant imbalance in bandwidth allocation: the cross-DC flow's throughput is about 1.3 times that of the intra-DC flow. Besides, GEMINI's performance is significantly influenced by its parameters and we show this through the experiments in Appendix VII. In TIMELY (with a high threshold set to 55ms), the intra-DC flow is almost starved. The performance of TIMELY is highly dependent on the setting of the threshold, but it is challenging to select an appropriate shared threshold for both flows. In contrast, the cross-DC flow of DCTCP is essentially starved due to the low rate increase frequency. Note that the performance of DCTCP is highly dependent on the choice of thresholds.

In Figure 10, we first start an intra-DC flow and a cross-DC flow with the same rate (50Gbps). Then, an intra-DC flow started at the line rate joins at 1s and leaves at 3s. Gear (Figure 10(e)) stably shared bandwidth between intra-DC flow and cross-DC flow at 0-1s, and when the second intra-DC flow was added, all three flows quickly adjusted their rates to a fair position. We can observe from Figure 10(a) that when a new flows joins in, only the intra-DC flows respond to the available bandwidth variation. Similarly, in GEMINI, the cross-DC flow is insensitive to the addition of a new flow. In Figure 10(c), the cross-DC flow aggressively grab about 80% of the bandwidth, while the intra-DC flows shares the remaining bandwidth share. Annulus (Figure 10(d)) achieved a relatively fair distribution of bandwidth among three flows, but the throughput of each flow exhibited significant fluctuations.

2) Return to Data Center Flows: Considering that Gear flows estimate the bandwidth of cross-DC flows when they perform MI adjustments, we investigate whether the estimation operations will affect the performance if there is only intra-DC traffic. Figure 11 illustrates the throughput of flows with different algorithms when only DC traffic is present. A new intra-DC flow is started every second in 0-3s and an intra-DC flow departs every second within the interval of 4-6s. The experiment results show that all the algorithms show good fairness among flows when there are only intra-DC flows. Comparing Figure 11(a), Figure 11(b), and Figure 11(e), we find that Gear exhibits slightly higher throughput oscillations compared to HPCC and GEMINI. This results from the computational errors, *i.e.*, Gear flows may mistakenly assume the presence of cross-DC traffic, thereby leading to the oscillations of the sending rate. However, we can observe that the extra bandwidth oscillation incurred by Gear is limited and Gear flows can achieve quick convergence. DCTCP and Annulus



(a) HPCC-10Mbps (b) Gear-10Mbps (c) HPCC-50Gbps (d) Gear-50Gbps (e) HPCC-66Gbps (f) Gear-66Gbps

Fig. 12. The throughput of the intra-DC and cross-DC flows under the initial rates of 10Mbps, 50Gbps, and 66Gbps.

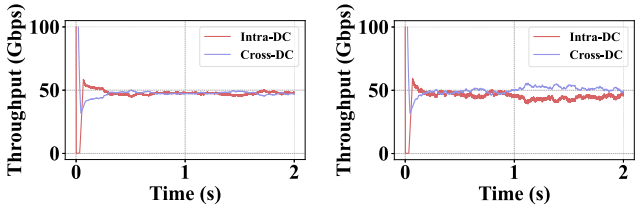
exhibit significant oscillations, which can be attributed to the proportional relationship between the AI factor and RTT.

3) Influence of the Initial Rate: During our experimentation, we observed that the initiation rate of flows significantly impacts the convergence behavior of different algorithms. In this section, we choose algorithms demonstrating convergence in line rate-started experiments: HPCC and Gear. We further investigate whether these algorithms could converge to fair bandwidth share under varying initial rates: [10Mbps, 50Gbps, 66Gbps].

Figure 12 presents the performance of HPCC and Gear with initial rates set to 10Mbps (minor rate), 50Gbps (correct rate), or 66Gbps (cause slight congestion). We do not choose 100Gbps because the experiment with an initial rate of 100Gbps is shown in Figure 9. When the rate is set to 10Mbps, cross-DC flow in HPCC (Figure 12(a)) struggles to obtain any bandwidth share. This results from the rapid responses of intra-DC flows, which quickly consume the available bandwidth and raise the link utilization. In contrast, Gear employs two mechanisms to ensure the two flows move toward the fairness point. Firstly, the cross-DC flow doubles the rate adjustment when performing MI operations, attempting to reclaim bandwidth from intra-DC flows. Secondly, the coefficient associated with the fairness factor (W_{AI}) for cross-DC flows ensures its competitiveness during additive operations.

When the initial rate is set to 50Gbps, the link utilization slightly exceeds the target utilization ($\eta = 95\%$). HPCC's intra-DC flow (Figure 12(c)) reduces their rates to drive the link utilization towards the target position, while the cross-DC flow maintains a relati vely stable rate throughout the process. In contrast, Gear's cross-DC flow attempts to compensate for the intra-DC flows and adjusts its rate accordingly. Figure 12(d) demonstrates that, even with a small disparity in rates, both the intra-DC and cross-DC flows progressively converge towards fairness.

Under an initial rate of 66Gbps, both the HPCC and Gear flow converge towards fairness. However, it is unfortunate that the convergence speed of HPCC is too slow that it does not converge even until 2 seconds (in Figure 12(e)). The reason behind this is that, after the link utilization approaches the target, HPCC relies on additive increase to create congestion and then performs multiplicative decrease to promote fairness. On the other hand, Gear maintains a fast convergence speed (in Figure 12(f)). Both flows have already reached a fairly balanced state by 0.5 seconds.



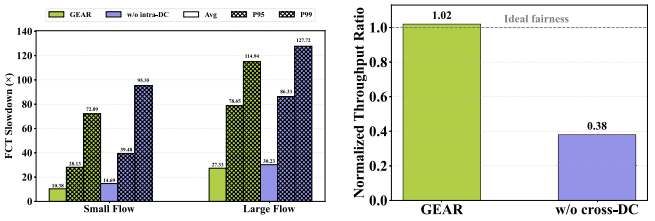
(a) All INT entries (b) INT entry replacement

Fig. 13. Different INT entry collection strategies.

4) The Impact of Int Entry Replacement: In this section, we discuss the impact of different INT entry replacement strategies on performance. In previous works, the bottleneck was often identified by directly selecting the INT entry with the maximum queue length (qlen) [11]. However, utilization is influenced by factors beyond just qlen. Packets can temporarily fill the buffer in the face of traffic bursts, but this does not necessarily indicate a high port utilization over time. The sender underestimates the congestion level of the network if it fails to locate the most congested port. This leads to an insufficient reduction or even an increase in the sending rate. To investigate this further, we will conduct experiments under two scenarios: 1) retaining all INT entries and computing the utilization of all passed ports, 2) just retaining the INT entry with the maximum qlen (if two INT entries have the same qlen, we will randomly select one of them). We use the same experimental setup as in Section IV-B. We observe the throughput of one intra-DC flow and one cross-DC flow that were initiated simultaneously and experienced the same upstream path within the data center.

The experiment result is shown in Figure 13. When we retain all INT entries (Figure 13(a)), the sender can accurately identify the most congested port, allowing both cross-DC and intra-DC flows to quickly converge to a fair position. Due to the presence of background traffic, throughput may experience slight fluctuations. Once we replace the INT entry just according to the qlen (Figure 13), the endpoint may not capture the most congested point. Since intra-DC flows receive INT information more frequently, they can still correct their behavior based on the subsequent accurate information. In contrast, cross-DC flows adjust infrequently, and if they respond to incorrect congestion information, their compensation for intra-DC flows decreases. Therefore, cross-DC flow takes more bandwidth than intra-DC flow. Nevertheless, most of the time, the error rate is relatively low; therefore, this mechanism does not result in significant throughput differences.

5) Ablation Experiments: Figure 14 presents an ablation study that isolates the contributions of the intra-DC and cross-DC logic in GEAR. In this experiment, we selectively replace either component with the HPCC baseline, while keeping all



(a) FCT Slowdown Comparison (GEAR vs. GEAR w/o Intra-DC Logic)

(b) Normalized Throughput Ratio (GEAR vs. GEAR w/o Cross-DC Logic)

Fig. 14. Ablation Testing of GEAR Architecture: (a) FCT Slowdown Comparison When Intra-DC Logic Is Removed; (b) Fairness Degradation When Cross-DC Logic Is Removed.

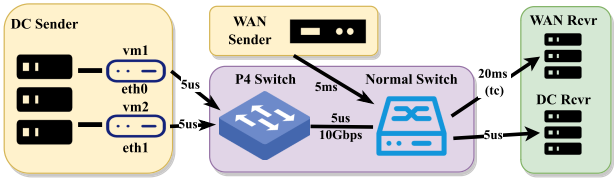


Fig. 15. The topology of the testbed.

other settings identical to those in Sections IV-B (for intra-DC ablation) and IV.A (for cross-DC ablation).

In particular, when the intra-DC logic is removed (w/o intra-DC), GEAR can no longer accurately estimate the bandwidth share of cross-DC flows at the bottleneck, and thus fails to promptly drain queues under congestion. This leads to a marked increase in tail latency. As shown in Figure 14(a), the 99 percentile FCT inflation for small flows rises from 72.09× (full GEAR) to 95.35×, and for large flows from 114.94× to 127.72×; average and 95 percentile FCT also increase substantially.

By contrast, when the cross-DC logic is removed (w/o cross-DC), intra-DC flows still adjust link utilization correctly, but cross-DC flows no longer perform fairness rate adjustments. This causes slow fairness convergence and severe imbalance in throughput. As depicted in Figure 14(b), the throughput ratio of intra-DC to cross-DC flows drops to 0.38, compared to 1.02 under the full GEAR design, which is close to ideal fairness.

D. Prototype-Based Experiment

As Figure 15 shows, we build a small testbed with four Intel Xeon (R) servers (with kernel version 4.15.0) and Intel (R) Celeron (R) Tofino P4 switch (with kernel version 3.16.39-OpenNetworkLinux). We employ two virtual machines (VMs) in the sender to start two flows simultaneously via different NICs. VM1 sends data to a server in the WAN (WAN Rcvr), while VM2 sends data to a server within the data center (DC Rcvr). The base RTT for each hop is approximately 5 microseconds. To emulate the latency of the WAN, we set the delay between the normal switch and the WAN receiver to 20ms through traffic control (TC). The egress bandwidth of the P4 switch is set to 10Gbps, serving as the shared bottleneck for both flows. In our prototype implementation, we select DCTCP, TIMELY, HPCC, and GEMINI as comparison algorithms and implement these protocols using DPDK.

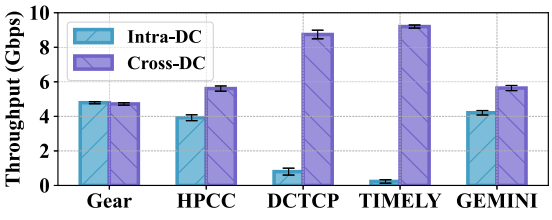


Fig. 16. The throughput of intra-DC and cross-DC flows in the testbed.

In this small topology, we conducted two experiments: 1) to verify that cross-DC flows and intra-DC flows can evenly share the bottleneck bandwidth. 2) to confirm that when congestion is located at the WAN, bandwidth can still be properly allocated to flows. We repeated each experiment 30 times to observe their performance. For the **first** experiment, vm1 and vm2 start two flows at the line rate (10Gbps), respectively, and record the average throughput of both flows over 30 seconds. Figure 16 shows that the experiment results align with our simulation experiments, where the error bars indicate the measurement jitter. We can observe that Gear significantly enhances fairness between intra-DC flows and cross-DC flows. Both flow types achieve nearly identical bandwidth shares with minimal jitter. In all other algorithms, intra-flow experiences throughput degradation, with DCTCP and TIMELY exhibiting more pronounced effects.

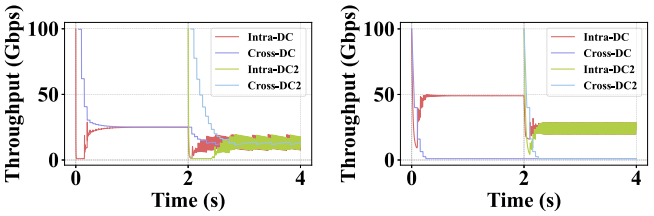
We are concerned that adjustments based on INT information might prevent flows from properly responding to congestion in the WAN. Therefore, we first let the WAN sender start a 2Gbps background flow to the WAN Recr and set the maximum receiving rate of the WAN Recr to 5Gbps. Then, the tested flows from vm1 and vm2 are started. In this setup, the ideal bandwidth ratio is: intra-DC flow: cross-DC flow = 7:3. Under this configuration, all algorithms can converge near this ratio within 5 seconds, as we limit the more competitive cross-DC flow. However, there are significant differences in the convergence times of the various algorithms. Since DCTCP, TIMELY, and GEMINI increase their cwnds in an additive manner, the intra-DC flow takes nearly 5 seconds to occupy the bandwidth vacated by the cross-DC flow. This results in impairments in link utilization. In contrast, Gear and HPCC complete convergence within 100 ms.

V. Discussion

A. Deployment Considerations for Int-Based Mechanisms at Cross Datacenter Scale

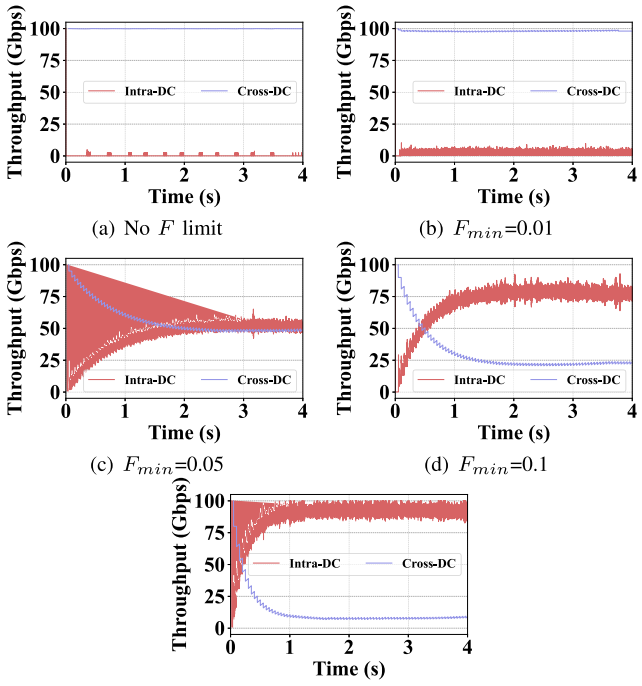
To evaluate the performance of Gear under a partial INT deployment, we build a topology consisting of three racks, each containing two hosts. Rack A and Rack B are located within the same datacenter, while Rack C is situated in a remote datacenter. All racks are interconnected through a core switch. In this experiment, only the ToR switches support INT.

We first launch four flows from Rack A (two intra-DC and two cross-DC), followed by four additional flows (two intra-DC and two cross-DC) from Rack B at 2 seconds. Before 2s, the bottleneck is located at the ToR switch of Rack A that supports INT; after 2s, the bottleneck shifts to the core switch, which doesn't support INT.



(a) Brownfield Gear (b) Brownfield Poseidon

Fig. 17. Performance of GEAR and Poseidon under a partial INT deployment. Only ToR switches support INT, while the core switch does not. (a) GEAR maintains rapid convergence when congestion occurs at INT-enabled ToR switches and still achieves eventual convergence when the bottleneck shifts to the non-INT core switch. (b) Poseidon suffers from severe cross-DC flow starvation due to excessive delay signals caused by RTT heterogeneity.



(a) No F limit (b) F_{min}=0.01

(c) F_{min}=0.05 (d) F_{min}=0.1

(e) F_{min}=0.2

Fig. 18. The performance of GEMINI with different F-limit.

For comparison, we implement Poseidon's strategy for partial INT deployments. According to Poseidon's design, congestion at non-INT switches is inferred using the difference between the end-to-end fabric delay and the maximum single-hop delay(MPD) reported by INT-enabled switches (i.e., RTT - MPD). The CC decisions are then based on the maximum value between RTT - MPD and MPD.

As shown in Figure 17(b), Poseidon exhibits severe starvation for cross-DC flows. The root cause is the significant RTT disparity between cross-DC and intra-DC flows. Specifically, Poseidon's RTT-MPD mechanism inherently assumes the fabric delay and MPD are of similar magnitude. However, in cross datacenter networks, the end-to-end fabric delay is typically several orders of magnitude greater than single-hop queuing delays. Thus, cross-DC flows consistently observe excessively large delay signals. This misinterpretation causes Poseidon to aggressively reduce sending rates, ultimately causing severe performance degradation or starvation of cross-DC traffic.

In contrast, as illustrated in Figure 17(a), Gear achieves rapid and effective convergence between 0s and 2s, when congestion occurs at INT-enabled ToR switches. Gear utilizes INT telemetry from the most congested hop along the flow path, effectively capturing critical congestion information. Given that most congestion events within a datacenter occur at the ToR switches, Gear retains most of its performance benefits even in a partially deployed INT scenario.

After 2s, when congestion shifts to the core switch, which doesn't support INT, Gear experiences some performance fluctuations yet still achieves eventual convergence. This is because congestion at non-INT switches triggers PFC, exerting backpressure on upstream ToR switches. As a result, the upstream ToR switches pause data transmission downstream and accumulate queues. Gear can still indirectly infer congestion conditions through this delayed congestion signal provided by ToR switches. Although this indirect signal introduces latency and oscillation during the convergence process, Gear stabilizes at a reasonable state.

It is worth noting that existing INT-based algorithms, such as HPCC, encounter similar challenges in partial deployment at cross datacenter scale. Our future research will continue to explore more effective solutions for deploying INT-based congestion control mechanisms in cross datacenter networks under partial deployment.

B. Analysis of Asynchronous Flow Start Effects

In Section III-C, to simplify the model, Gear assumes that all flows receive the INT information simultaneously and adjust their rates accordingly (as stated in Equation 5). However, flows may start at arbitrary times and, consequently, receive INT information at any point within an RTT. Therefore, in this section, we develop a mathematical model to account for scenarios where flows are initiated at different times.

Considering that, regardless of when flows start, they will receive INT information at least once within an RTT, we assume that there are n intra-DC flows: $a = (a_1, a_2, \ldots, a_n)$, and these flows undergo MI adjustments asynchronously at time $(t,\ t+k_2,\ \ldots,\ t+k_n)$. Then we can use the matrix to represent the relationship between the adjustment times and the changes in link utilization as follows (temporarily ignoring the AI factor):

$$H = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ \frac{\eta}{u(t)} & 1 & 1 & \cdots & 1 & 1 \\ \frac{\eta}{u(t)} & \frac{\eta}{u(t+k_2)} & 1 & \cdots & 1 & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\eta}{u(t)} & \frac{\eta}{u(t+k_2)} & \frac{\eta}{u(t+k_3)} & \cdots & \frac{\eta}{u(t+k_n)} & 1 \end{bmatrix}$$

$$H \cdot \begin{bmatrix} w^c_{a_1} \\ w^c_{a_2} \\ w^c_{a_3} \\ \vdots \\ w^c_{a_n} \\ w^c_b \end{bmatrix} = \begin{bmatrix} u(t) \\ u(t+k_2) \\ u(t+k_3) \\ \vdots \\ u(t+k_n) \\ u(t+T) \end{bmatrix} \cdot bw \tag{11}$$

$$0 < k_2 < k_3 < \ldots < k_n \leq T - IRTT$$

However, as the algorithm is distributed, each flow can only make decisions based on locally observed information without access to global data. For example, from the perspective of the flow a_1, it only has access to $u(t)$ and $u(t+T)$ so that a_1 cannot directly determine the value of w_b^c from Equation 11 (the number of unknowns exceeds the rank of the coefficient matrix) to guide the MI operation at time $t+T$. This implies that an exact solution to Equation 11 cannot be derived from the available information. Therefore, we attempt to simplify the equation to improve the accuracy of the inference for w_a and w_b. From the perspective of each flow, data center flows are divided into the flow itself and other intra-data center flows.

We denote the flow itself by a_1, other intra-data center flows by a_{2-n}, and cross-data center flows by b. Equation 11 can therefore be reformulated as follows:

$$\begin{bmatrix} 1 & 1 & 1 \\ \frac{\eta}{u(t)} & 1 & 1 \\ \frac{\eta}{u(t)} & \frac{\eta}{\boldsymbol{U}} & 1 \end{bmatrix} \begin{bmatrix} w_{a_1}^c \\ w_{a_{2-n}}^c \\ w_b^c \end{bmatrix} = \begin{bmatrix} u(t) \cdot bw \\ \boldsymbol{U} \cdot bw \\ u(t+T) \cdot bw \end{bmatrix} \tag{12}$$

where $\frac{\eta}{\boldsymbol{U}} \cdot w_{a_{2-n}}^c = \sum_{i=2}^{n} \frac{\eta}{u(t+k_i)} w_{a_i}^c$.

As every intermediate state of link utilization ($u(t+k_2)$ to $u(t+k_n)$) is inaccessible, we cannot determine the exact value of U. Instead, we can provide the range of U:

$$U \in [min(u(t), u(t+T)),\ max(u(t), u(t+T))]$$

Here, we set $U = u(t+T)$ and take this into equation 12, then $w_{a_{2-n}}^c$ and w_b^c can be easily calculated. The reasons for setting $U = u(t+T)$ are as follows:

- When $u(t) < \eta$, the link utilization gradually increases as time progresses, we have:

  $$U \in [u(t),\ u(t+T)]$$

  In this case, $U = u(t)$ will result in an underestimation of $w_{a_{2-n}}^c$ and an overestimation of w_b^c. Conversely, $U = u(t+T)$ will lead to an underestimation of w_b^c. To avoid congestion on the link, we choose to set $U = u(t+T)$.
- When $u(t) > \eta$, we have:

  $$U \in [u(t+T),\ u(t)]$$

  In this case, $U = u(t)$ and $U = u(t+T)$ will result in underestimation and overestimation of w_b^c, respectively. To rapidly alleviate congestion, we still choose to set $U = u(t+T)$.

VI. Related Work

CC is a highly popular topic. In this section, we try to cover several works that are highly related to Gear.

Congestion Control with buffer-based signals. The majority of CC algorithms rely on buffer-based signals such as RTT and ECN, as they enable senders to regulate queue length and delay more effectively. RTT-based algorithms include Vegas [14], FastTCP [16], TIMELY [9], and Swift [10], wherein [9] and [10] are deployed in DC and require hardware assistance for high precision RTT measurements. ECN-based algorithms are represented by the window-based DCTCP [5] and rate-based [6]. However, (*i*), senders can only converge iteratively to the available bandwidth. (*ii*) the parameter settings face the dilemma of simultaneously satisfying the low-latency requirement of intra-DC flows and high-throughput demand of cross-DC flows. Therefore, we tend to select a congestion control signal with parameter settings that are fair to both cross-DC and intra-DC flows.

Congestion Control with INT information. Since the inception of INT [36] technology, research and innovation in INT-based CC [7], [11], [12], [37], [38], [39], [40] have been developing. In HPCC [7], INT was first applied to CC within DCN to achieve precise rate control and rapid convergence. Poseidon [11] and PowerTCP [12] both utilize INT information to identify the bottleneck switch as the one with the longest queuing time. Poseidon sets lower queueing delay thresholds for high-speed flows to achieve fair bandwidth allocation. At the same time, PowerTCP comprehensively considers the queuing delay and delay gradient at the bottleneck to determine the severity of congestion. These algorithms are well suited for DCN but struggle to balance flow fairness under cross-DCN scenarios.

Congestion Control for Cross-DCNs. Previous researchers have investigated CC algorithms for cross-DCN. Their approaches can be categorized into two types. The first type utilizes the forward feedback mechanism to create control loops with the same length. For instance, when the bottleneck lies in DC, Annulus [20] treats the TOR switch in DC as the congestion point, which directly forwards the QCN packets back to the host. The second type applies distinct control mechanisms for flows with different control loop lengths: GEMINI employs the delay signal to limit the cross-DC delay and utilizes ECN to mitigate intra-DC packet loss. Besides, GEMINI [22] adjusts the rate update step based on the length of RTTs. GTCP [31] employs a receiver-driven CC inside the DCN while a sender-driven protocol for the WAN network. When congestion is detected within the DC, the cross-DC flow switches to a receiver-driven mode to mitigate any adverse effects on intra-DC flows. Otherwise, the sender is responsible for detecting the available bandwidth. These algorithms aim to achieve efficient and fair source allocation for intra- and cross-DC traffic. However, as previously analyzed, they still employ buffer-based signals, which make it difficult to set parameters that satisfy the fair goal of both types of flows.

VII. Conclusion

Intra-DC and cross-DC traffic show significantly distinct characteristics, thereby proposing different requirements for CC algorithms. We show that current CC algorithms are designed for individual networks and ill-suited for cross-DCN. This paper introduces Gear, a CC model designed for cross-DCN. In Gear, intra-DC flows are responsible for rapidly converging to available bandwidth, and cross-DC flows proactively control bandwidth allocation to ensure fairness. Our experiment results show that Gear greatly improves the fairness between intra-DC and cross-DC flows. In the meantime, Gear also achieves high throughput and low latency simultaneously. We envision that this work will motivate further investigation of the usage of INT information under cross-DCN scenarios.

APPENDIX
THE PERFORMANCE OF GEMINI WITH DIFFERENT F LIMIT

In the paper of GEMINI, the parameter H is used to limit the growth of the cwnd of cross-DC flows, restricting the cwnd increase within the range of $[0.1,\ 5]$. GEMINI runs the experiments with 1Gbps or 10Gbps links. However, when the link capacity increases to 100Gbps, we observe that the limitation imposed by H will result in the starvation of cross-DC flows. Additionally, when there is a significant difference in RTTs between intra-DC flow and cross-DC flow (*e.g.*, 100x), the scale factor F becomes too small so that cross-DC flows can hardly reduce their rates during congestion. Therefore, in our experiments, we remove the limitation by H and give a limitation to the scale factor F.

The selection of parameter F has a substantial impact on the experiment results, as evidenced by Figure 18. Without imposing any restrictions on F or allowing the minimum value of F to be as low as 0.01, the intra-DC flows struggle to obtain bandwidth allocation. At the minimum value of F = 0.05 (Figure 18(c)), the intra-DC and cross-DC flows tend towards fairness. Increasing the minimum value of F enhances the competitiveness of the intra-DC flow (in Figure 18(d) and Figure 18(e)). Our experiments in this paper are conducted with $F_min = 0.05$. The experiment shows that GEMINI has a strong dependence on parameter selection and requires parameter tuning in new environments.

REFERENCES

[1] *Amazing Cloud Adoption Statistics [2023]: Cloud Migration, Computing, and More*. 2022. Accessed: 2025. [Online]. Available: https://www.zippia.com/advice/cloud-adoption-statistics/

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

[3] C.-Y. Hong et al., "B4 and after: Managing hierarchy, partitioning, and asymmetry for availability and scale in Google's software-defined WAN," in *Proc. Conf. ACM Special Interest Group Data Commun.*, Aug. 2018, pp. 74–87.

[4] D. Kamińska et al., "Virtual reality and its applications in education: Survey," *Information*, vol. 10, no. 10, p. 318, Oct. 2019.

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

[6] 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.

[7] Y. Li et al., "HPCC: High precision congestion control," in *Proc. ACM Special Interest Group Data Commun.*, Aug. 2019, pp. 44–58.

[8] W. Cheng, K. Qian, W. Jiang, T. Zhang, and F. Ren, "Re-architecting congestion management in lossless Ethernet," in *Proc. 17th USENIX Symp. Networked Syst. Design Implement.*, 2020, pp. 19–36.

[9] R. Mittal et al., "TIMELY: RTT-based congestion control for the datacenter," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 537–550, Sep. 2015.

[10] G. Kumar et al., "Swift: Delay is simple and effective for congestion control in the datacenter," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., architectures, protocols Comput. Commun.*, Jul. 2020, pp. 514–528.

[11] W. Wang et al., "Poseidon: Efficient, robust, and practical datacenter CC via deployable INT," in *Proc. 20th USENIX NSDI*, 2023, pp. 255–274.

[12] V. Addanki, O. Michel, and S. Schmid, "PowerTCP: Pushing the performance limits of datacenter networks," in *Proc. 19th USENIX NSDI*, 2021, pp. 51–70.

[13] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *ACM SIGOPS Operating Syst. Rev.*, vol. 42, no. 5, pp. 64–74, Jul. 2008.

[14] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," in *Proc. Conf. Commun. Archit., Protocols Appl.*, 1994, pp. 24–35.

[15] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time," *Queue*, vol. 14, no. 5, pp. 20–53, Oct. 2016.

[16] C. Jin, D. X. Wei, and S. H. Low, "FAST TCP: Motivation, architecture, algorithms, performance," in *Proc. IEEE INFOCOM*, vol. 4, May 2004, pp. 2490–2501.

[17] V. Arun and H. Balakrishnan, "Copa: Practical delay-based congestion control for the Internet," in *Proc. USENIX Symp. Netw. Syst. Design Implement.*, 2018, pp. 329–342.

[18] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira, "PCC: Re-architecting congestion control for consistent high performance," in *Proc. 12th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2015, pp. 395–408.

[19] M. Don et al., "PCC Vivace: Online-learning congestion control," in *Proc. 15th USENIX Symp. Networked Syst. Design Implement.*, 2018, pp. 343–356.

[20] A. Saeed et al., "Annulus: A dual congestion control loop for datacenter and WAN traffic aggregates," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Archit., Protocols Comput. Commun.*, Jul. 2020, pp. 735–749.

[21] M. Khosroshahy, "Congestion avoidance and control," Tech. Rep., Jan. 2009. Accessed: 2025. [Online]. Available: https://www.analytical.works/Tech-Report-Congestion-Avoidance-and-Control.html

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

[23] *In-Band Network Telemetry—A Powerful Analytics Framework for Your Data Center*. Open Compute Project Foundation. Mar. 2018. Accessed: 2025. [Online]. Available: https://www.opencompute.org/

[24] *In-Band Network Telemetry in Broadcom Tomahawk 3*. Broadcom Inc. Oct. 2018. Accessed: 2025. [Online]. Available: https://www.broadcom.com/company/news/productreleases/41086

[25] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 281–292, Aug. 2004.

[26] S. M. Rumble, D. Ongaro, R. Stutsman, M. Rosenblum, and J. K. Ousterhout, "It's time for low latency," in *Proc. 13th Workshop HotOS XIII*, 2011, pp. 1–5.

[27] *A. Production*, 2019. Accessed: 2025. [Online]. Available: https://www.arista.com/en/products

[28] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proc. HotCloud*, Feb. 2010, pp. 1–10.

[29] D. Borthakur, "The Hadoop distributed file system: Architecture and design," Hadoop Project Website, Tech. Rep., 2007. Accessed: 2025. [Online]. Available: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

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

[31] S. Zou, J. Huang, J. Liu, T. Zhang, N. Jiang, and J. Wang, "GTCP: Hybrid congestion control for cross-datacenter networks," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2021, pp. 932–942.

[32] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *Proc. 2015 ACM SIGCOMM*, 2015, pp. 123–137.

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

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

[35] C. E. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing," in *Proc. Int. Conf. Parallel Process.*, 1994, pp. 170–179.

[36] C. Kim et al., "In-band network telemetry via programmable dataplanes," in *Proc. ACM SIGCOMM*, 2015, pp. 1–2.

[37] R. Ben Basat, S. Ramanathan, Y. Li, G. Antichi, M. Yu, and M. Mitzenmacher, "PINT: Probabilistic in-band network telemetry," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Archit., Protocols Comput. Commun.*, Jul. 2020, pp. 662–680.

[38] P. Zhang, H. Zhang, Y. Dai, Y. Pi, J. Wang, and J. Liao, "Cache-INT: In-network caching-enabled in-band network telemetry," *Comput. Netw.*, vol. 269, Sep. 2025, Art. no. 111404.
[39] K. Yang et al., "SketchINT: Empowering INT with TowerSketch for per-flow per-switch measurement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 11, pp. 2876–2894, Nov. 2023.
[40] X. Jie, J. Han, G. Chen, H. Wang, P. Hong, and K. Xue, "CACC: A congestion-aware control mechanism to reduce INT overhead and PFC pause delay," *IEEE Trans. Netw. Service Manage.*, vol. 21, no. 6, pp. 6382–6397, Dec. 2024.

Feixue Han received the B.S. degree from Beihang University, Beijing, China, in 2018. She is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Tsinghua University, Beijing. Her current research interests include congestion control and traffic scheduling in heterogeneous network scenarios.

Qing Li (Senior Member, IEEE) received the B.S. degree from Dalian University of Technology, Dalian, China, and the Ph.D. degree from Tsinghua University, Beijing, China. He is currently an Associate Researcher with Peng Cheng Laboratory, Shenzhen, China. His research interests include reliable and scalable routing of the internet, intelligent self-running network, and edge computing.

Rui Huang received the B.S. degree in software engineering from Beihang University in 2024. He is currently pursuing the master's degree with the Department of Computer Science and Technology, Tsinghua University, under the supervision of a Professor Yong Jiang from Tsinghua University and an Associate Researcher Qing Li from Peng Cheng Laboratory. His research interests primarily focus on data center networks.

Huiling Jiang received the master's degree in computer science from Tsinghua University. She works with Alibaba Cloud, mainly focusing on the development of integrated software and hardware gateways, which mainly involves optimizing public networks, iterating and upgrading network architectures, and monitoring and managing network quality.

Yong Jiang (Member, IEEE) received the B.S. and Ph.D. degrees in computer science and technology from Tsinghua University, Beijing, China. He is currently a Full Professor with Tsinghua Shenzhen International Graduate School. His research interests include the future network architecture, the Internet QoS, and network function virtualization.