ECN Works Better for Selective Dropping in High Bandwidth-Delay Product Connections

Mahmoud Bahnasy and Ali Munir
Huawei Technologies Ltd.
Waterloo, Canada
e-mail: ali.munir@huawei.com

Junjie Niu, Zhibo Yan, and Peng Dong
Huawei Technologies Ltd.
Nanjing, China

Yashar Ganjali
University of Toronto.
Toronto, Canada
e-mail: ganjali7@cs.toronto.edu

***Abstract*—Modern Internet services and applications distribute data and compute on different geographically distributed data centers to improve performance and reliability. For example, content providers distribute their data centers among different regional areas and periodically replicate the content between data centers. Moreover, Geo-Distributed Machine Learning (Geo-DML) is emerged to satisfy the need to train large sophisticated models. Such a new model of training requires a very robust and reliable inter-data center transport layer. Such a design raises the need for high-speed reliable inter-data center transport. The conservative reaction of the Transport Control Protocol (TCP) in such long Round-Trip Time (RTT) connections cause huge degradation in the throughput. For example, TCP is able to utilize only 40% of a 10-Gbps link between two data centers with $RTT = 10ms$ and it gets even worse when packet loss occurs. We present Data Center Interconnect-Bridging (DCIB) as a method for recovering lost packets in such a scenario by allowing the router to drop pre-selected packets that can easily recovered by the edge routers without triggering TCP's reaction at the host. Several simulation experiments show that DCIB can increase link utilization by a factor of 6x in highly congested connections, and reduce flow completion time by up to 85%.**

Keywords-Inter-Data Centers Communication; Cross Data Centers Communications; Transport Protocol; Congestion Control.

I. INTRODUCTION

Data Center Interconnect (DCI) technology connects two or more data centers together over short, medium or long distances using high-speed leased lines or the Wide Area Network (WAN) (Figure 1). Data Centers (DC) often serve many geo-distributed applications requiring huge amounts of data transfers between the data centers. For example, in applications such as astronomy, storage nodes are often separated from compute nodes in geographically distributed data centers. Such an approach allows astronomers to collect data close to the origin where telescopes are located while transferring the data to remote data centers for low-cost processing. The performance of the inter-DC traffic relies on the quality of the network path and the underlying transport protocol.

DCI imposes significant challenges in designing the transport layer because of the distinct characteristics of the routers and switches along the datapath. First, there is no administrative control available at the switches and routers along the path. Therefore it is hard to provide any performance guarantees. Second, bursty behavior of Transmission Control Protocol (TCP) and synchronization between flows can cause packet loss with no congestion (i.e., false congestion signal)

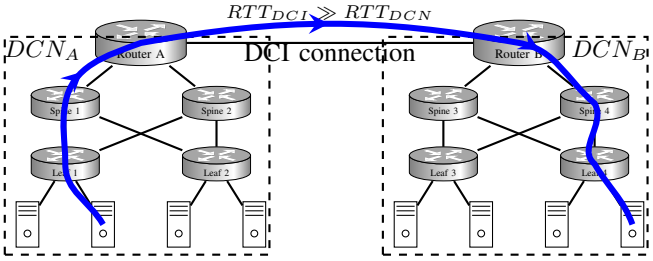


Figure 1. A typical DCI topology

which triggers rate reduction that takes a few Round Trip Time (RTT)s to recover. Finally, there is no explicit congestion signal available. Hence, it is very difficult to know the location and reason for poor performance in the network.

Existing congestion control mechanisms leverage packet loss [1], Explicit Congestion Notification (ECN) [2][3] or delay [**timely**][4], or a combination of them [5] as congestion signals. However, congestion signals, such as ECN and RTT, get delayed because of the long latency of DCI connections. Moreover, traditional TCP reaction to packet loss significantly degrades the application performance. For example, TCP may react aggressively to packet loss, yet at the same time it probes for available bandwidth very carefully to minimize packet loss in the network. Because of the large Bandwidth-Delay Product (BDP), TCP reacts slowly to congestion, and it recovers slowly available bandwidth which hinder achieving full link utilization. For that reason, DCI links are usually underutilized and can only achieve less than 40% of its link capacity, on average, even with zero packet loss (Review Figure 4).

In this work, we provide a method, Data Center Interconnect-Bridging (DCIB), to minimize the impact of packet loss on TCP, to maintain high throughput, and to achieve higher link utilization. DCIB is not a new congestion control mechanism. In fact, it can be considered as a plugable that can be integrated into any TCP protocol. The key idea here is to handle packet loss and packet retransmission inside the network itself, without triggering rate reduction and packet retransmission at the host, which results in throughput degradation. To that end, DCIB adds two main features to the edge routers. First, it selectively marks the ECN bit of the outgoing packets. The routers along the DCI path use this ECN mark for selective packet drop instead of randomly dropping

data packets. Although, we can't modify the programmability of routers across DCI path, DCIB only requires enabling a simple feature, such as ECN, which can be enabled as part of the SLA agreement. Second, the edge router is responsible for retransmitting lost packets. For this purpose, it stores the marked packets in a local buffer and then retransmits them upon detecting packet loss.

The main innovation is that DCIB presents a new way of using ECN, i.e., not to convey the congestion, but to inform routers to drop, in case of congestion, certain packets that can be recovered fast by the edge routers without triggering the conservative behavior of TCP at the end host. Hence, DCIB provides faster recovery from lost packets, while keeping the control loop short (between routers). This flow control mechanism aims to resolve transient congestion that is caused by flow synchronization and/or the bursty behavior of TCP. If edge routers can't recover packet loss either because the number of packet loss is high, or the congestion is long-lasting, normal TCP behavior is triggered at the hosts.

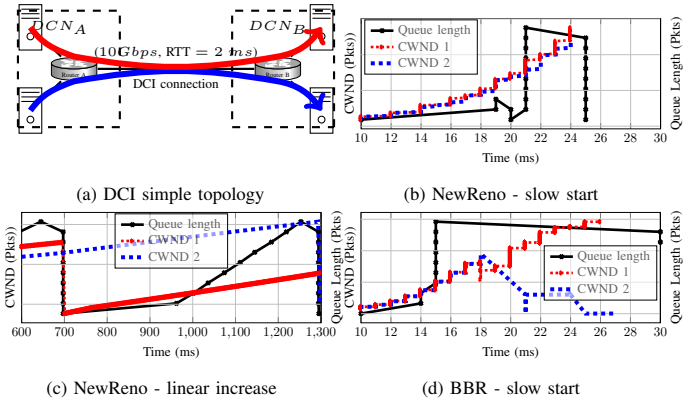
The rest of the paper is organized as follows; Section II presents the motivation behind this work. The design of our proposed protocol is discussed in Section III. Section IV includes the implementation details of our protocol and the experimental results. Several discussion points regarding the design of DCIB is discussed in Section V. We conclude the paper in Section VI.

II. Motivation

The challenge that congestion control mechanisms face in the large BDP scenario is the long control loop. Unfortunately, this issue is intrinsic to the physical property of the system. Hence, building a new congestion control based on traditional congestion signals, such as packet loss, delay, or ECN marking, is not going to help too much. For example in an ideal system where all network components participate in resolving congestion, when a switch detects congestion and starts signalling the congestion by ECN-marking data packets, assuming it knows exactly which flow is the culprit flow. Such a signal needs at least one RTT (i.e., a few ms) to convey the signal plus the reaction time the sender requires. One can conclude that by the time the host reacts to the received congestion signal, the congestion might have already been resolved. [6] shows experimentally that DCI average throughput drops by 18%-37% as buffer decreases. It also demonstrates that packet loss occurs in DCI reaches 5% for both TCP Cubic and Bottleneck Bandwidth and Round-trip propagation time (BBR) even with deep buffers (i.e., 50 MBytes).

A. Illustrative Example

To demonstrate the impact of unpredictable packet loss and delayed signal on TCP NewReno and BBR [5], we conduct a simulation using Ns-3 [7]. We consider two Data Center Network (DCN) networks connected using a dedicated link of capacity 10 $Gbps$ and latency 2 ms (Figure 2a). In this simulation, we start two long-lived flows from DCN_A towards DCN_B. Figure 2b shows the Congestion Window (CWND) and the queue length for the two protocols in this simple scenario. One can clearly see the slow start behavior of TCP NewReno where CWND is increased exponentially with every received ACK after one RTT (at $t = 10\ ms,\ 12\ ms, 14\ ms,\ ...\ etc$). Because RTT is larger than the time needed to transmit CWND worth of data, the two flows behave in an on-off manner. One can notice that although the queue reached the maximum capacity at $t = 21ms$ and a packet loss occurs, the two flows keep increasing their CWND with the arrival of every acknowledgment received in the next cycle and do not recognize the packet loss signal till $t > 25\ ms$.



(a) DCI simple topology (b) NewReno - slow start

(c) NewReno - linear increase (d) BBR - slow start

Figure 2. TCP reaction is slow and might unnecessarily reduce CWND even after congestion disappeared.

Similarly, Figure 2c depicts the same behavior when the two flows reach the end of the linear increase phase. It shows that Flow 1 (depicted in red) reacted to the first congestion at $t = 650\ ms$ after $\approx 50ms$ (i.e., 25 RTTs). The same occurs with Flow 2 at $t = 1250\ ms$ (depicted in blue).

Similar behavior was observed for the delay-based congestion control (BBR). Figure 2d demonstrates that BBR also reacts late to congestion signal and keeps increasing the CWND of the two flows even when the buffer size reaches the maximum allowed value at $t = 15ms$ causing higher packet loss rate. Figure 2d depicts that flow 1 and flow 2 do not start reducing their CWND till $t = 18\ ms$ and $t = 28\ ms$, respectively.

To understand the impact of large BDP on TCP delayed reaction we run the same experiment while increasing the DCI link latency (illustrated Figure 3 by RTT). One can notice that as the link latency increases, which increases BDP, the throughput of TCP protocol (i.e., NewReno, Cubic and BBR TCP) degrades drastically.

Furthermore, Figure 4 illustrate the impact of packet loss on TCP throughput. We vary the packet loss from 0‰ to 3‰ and plot the throughput of TCP NewReno, Cubic and BBR. We can observe that as the loss rate increases, the throughput drops exponentially. We conclude that TCP by itself can't maintain high throughput in such an environment because of the long control loop.

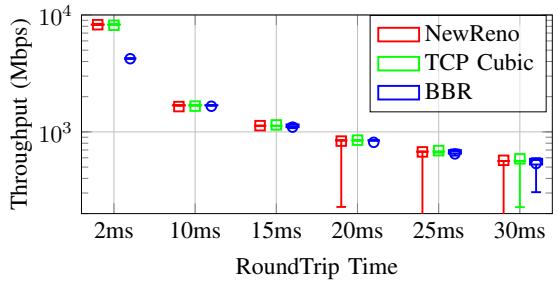


Figure 3. Large BDP greatly degrades the performance of TCP (The average is represented by circles).

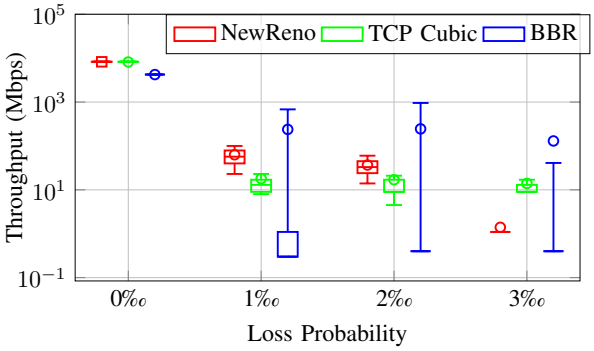


Figure 4. Packet loss on DCI link reduces throughput significantly (The average is represented by circles).

III. Design

We propose DCIB as a solution that helps mitigating the effect of transient congestion and packet loss while allowing TCP to react to long-lasting congestion.

The key idea behind DCIB is to handle packet loss and packet retransmission inside the network itself. To achieve this goal, DCIB leverages the existing feature in the routers and switches (i.e., ECN) for handling packet loss in the DCI, at the egress of the data center (at the border routers) for packet retransmission within the network. DCIB works as extension for any TCP to allow faster retransmission. Thus, hosts can maintain a high transmission rate while preventing rate reduction due to transient congestion. It also avoids waiting for three duplicate ACKs or the expiration of Retransmission Timeout (RTO) timer to trigger the retransmission which also accelerates the recovery process.

A. Selective Marking at the Source Edge Router

DCIB uses ECN for selective packet drop in case of congestion instead of congestion notification. DCIB allows edge routers to uniformly select a few packets that can be dropped in the network. The intuition here is that the edge routers can store and retransmit these packets if a packet drop is detected in the network. For this purpose, the ECN bit is set on all the outgoing packets except for the packets that are selected for drop. For flows going from router A to router B, router A selects or recommends the drop packets ratio within the range $[R - r, R]$ where R is the current transmission rate and r is the drop recommendation rate which is taken to be equal $0.1 \cdot R$ in our experiments (depicted by dotted area in Figure 5).

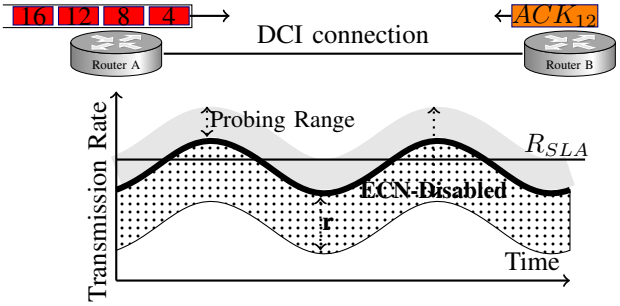


Figure 5. DCIB work mechanism

In addition, Router A clones these unmarked packets and stores them in a separate queue/buffer for retransmission in case of a packet loss. We call such a queue as stalled queue. Router A detect which packet was received/lost by comparing the received router-to-router ACK (explained later) with the head of the stalled queue. Therefore, Router A can retransmit packets, from the head of the stalled queue, for any packet for which an ACK is not received. We assume that there's one datapath between the two data centers. For multiple data paths, Router A can use one queue per path, however, we leave studying the effect of multiple data paths for future work. Figure 5 illustrate an example when Router A receives an ACK for packet 12, it indicates that all packets before packet 12 were lost, and must be retransmitted. Therefore, Router A starts retransmitting packets from the stalled queue till it reaches packet 12 which gets discarded (as it has been acknowledged) and the queue gets paused again until receiving another ACK. Such behavior can be carried out using PFC pause frames.

Moreover, Router A can probe for extra bandwidth by disabling ECN for few extra packets above the transmission rate (R) or the agreed-upon Service-Level Agreement (SLA) rate (R_{sla}) (depicted by gray area in Figure 5). These injected packets are used as probes which allows TCP to detect available bandwidth very quickly. In this paper, we did not consider changing TCP, hence, we leave that for future work.

B. Selective Packet Drop at Intermediate Routers

Intermediate routers between Router A and Router B perform the traditional Random Early Detection (RED) [8] process, or use Low Latency, Low Loss, and Scalable Throughput (L4S) [3] to drop ECN-disabled packets and mark for congestion ECN-enabled packets. Such a feature can be specified in the SLA agreement.

DCIB requires intermediate routers to drop selected packets in case of congestion which can be achieved by enabling RED [9] with ECN [10] or L4S [3].

We propose using such feature in a different way. We propose setting all packets to "ECN Capable" except selected packets that can be recovered at the edge router. Within the context of this paper, we also call the selected packets "ECN-Disabled" packets. In case of congestion (Average Qlen >

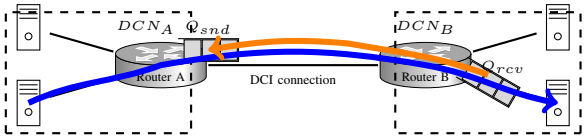


Figure 6. Queue Disciplines requirements

Threshold), The default behavior of RED is to mark "ECN Capable" packets for congestion and drop ECN-Disabled packets. By scarifying the selected packets that can be recovered fast at the edge router, we maintain stable throughput for TCP, hence, high resource utilization.

C. Router-to-Router ACK

At Router B, a Router-to-Router ACK is generated per ECN-disabled packet received. The intuition here is to act as a local receiver, and acknowledge a packet reception. Note that DCIB is designed to handle packet loss on the DCI links only, therefore we generate the ACKs at the destination edge router.

D. DCIB Internals

a) ***Maximum allowed Packet Drop Rate*** *(r):* The maximum packet drop can be calculated as $r \leq B/(\tau \times C)$, where B is dedicated stalled buffer capacity, τ is the inter-router round-trip time, and C is the link capacity. E.g., for a DCI link capacity of $C = 10\ Gbps$, $\tau = 10\ ms$ ($2,000Km$ distance) and edge routers' buffer sizes are $2\ MB$ each. We can calculate the maximum drop rate that DCIB mechanism can support as $r = (2e^6 \times 8bit)/(10e^{-3}s \times 10e^9 bps) = 0.16$. Hence, DCIB can recover up to 16% of packet loss without interrupting TCP protocol.

b) ***Queue Disciplines at the edge routers****:* Figure 6 shows the architecture of the Queue disciplines required at the edge routers. DCIB requires two types of queue disciplines; namely transmitting queue (Q_{snd}) and receiving queue (Q_{rcv}).

The algorithm of the sending process of Q_{snd} is illustrated in Figure 7. It calculates the selection rate (Line 1), modify ECN field in the IP header (Line 2), and clone the packet if selected (Lines 3-5). Figure 8 represents the receiving function at Q_{snd} (in Router A in our example). It reacts to received ACK from Router B by checking if it matches the head of the queue (Line 2). If a match is not found, it resumes transmission on the stalled queue until it reaches a packet that matches the received ACK (Lines 3-5). Otherwise, it drops the packet that matches the received ACK (Line 6). On the other hand, the process carried out at Router B, inside (Q_{rcv}), is depicted in Figure 9. Q_{rcv} verifies if the packet is selected for drop (ECN-disabled), it generates an ACK for each successfully received packet (Lines 1-3). Finally, it forwards the packet as normal towards its destination (Line 4).

IV. Evaluation

In this section, we illustrate the benefits of DCIB on transport performance using TCP Reno, TCP Cubic and BBR protocols.

A. DCIB with artificial loss:

In this experiment, we simulate two DCN network connected using a 10-Gbps DCI link. Link latency for all link inside each DCN is $5\mu s$. Link latency of the DCI link is $3\ ms$ which is equivalent to a 600-Km link between the two data centers. We start 15 long-lived flows from all hosts of DCN A towards DCN B. The simulation topology is depicted in Figure 10a. DCIB marks 10% of the traffic with ECN-disabled, and allows the rest to go through with ECN-enabled. In addition, we simulate congestion and packet loss in the WAN by artificially dropping 5% of the packets. We repeat the simulation twice, once with the assumption that WAN is not cooperative and they drop packets equally; i.e., 5% from both ECN-enabled and ECN-disabled traffic. In the second run, we imitate a cooperative WAN and drop 50% of the 10% ECN-disabled traffic only (i.e., 5%).

We demonstrate the performance by measuring the overall throughput at the DCI link. The results shown in Figure 10b and 10c illustrate the DCI throughput while using DCIB for both TCP Reno and TCP Cubic. It shows that DCIB enhances the performance by increasing the average throughput up to 6x for TCP Reno and up to 4.5x for TCP Cubic when packet drop takes place in ECN-disabled packets only. Moreover, when WAN drops packets regardless of the ECN marking, DCIB was able to enhance TCP throughput by 4x and 3.24x for TCP Reno and TCP Cubic, respectively. Figure 10d and 10e depict the same remarks by showing the Cumulative Distribution Function (CDF) of the DCI throughput.

B. DCIB with no artificial loss:

In this experiment we demonstrate the effect when packet loss only occurs in case of contention among high number flows on the limited buffer resources. To validate that, we repeat the previous simulation with the same number of long lived flows with 0% artificial drop probability. Figure 11a, 11b and 11c show that DCIB can greatly alleviate the effect of congestion-based packet loss for TCP Reno, Cubic and BBR, respectively. Even with a dedicated link for the DCI traffic, DCIB can enhance the average throughput up to 1.24x, 3.6x, and 2.4x for TCP Reno, Cubic and BBR, respectively. Although DCIB enhances the average and the median throughput for BBR by 1.85x and 2.4x, it did not enhance much the 99-

Data: Packet p
$r \leftarrow$ drop ratio;
Set $p.ECN$ on all packets except r percentage;
if $p.ECN! = 0$ **then**
  `/* Packet not selected for drop */`
  Clone packet;
  Store at stalled queue;
end
Transmit packet p;

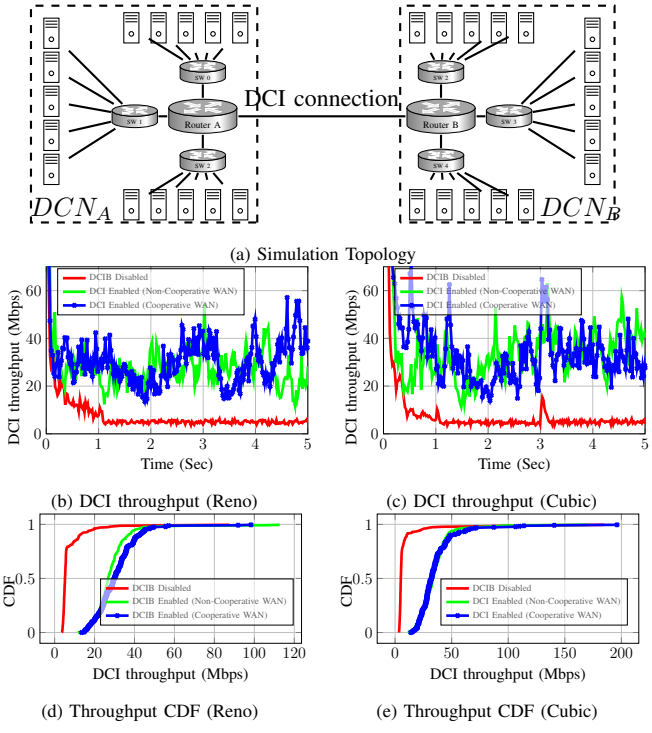
Figure 7. Packet processing at Q_{snd}

Data: Router-ot-Router ACK ack
Data: Stalled Queue $Q_{stalled}$
$head \leftarrow Q_{stalled}[0]$;
while $head.header! = ack.header$ **do**
    /* Received ACK does not match head of the queue */
    Transmit packet $head$;
    Wait for packet transmission;
    Drop $head$;
end
Drop $head$;
; // When ACK matches the head of the queue; drop the head of the queue
Return;

Figure 8. ACK processing at Q_{snd}

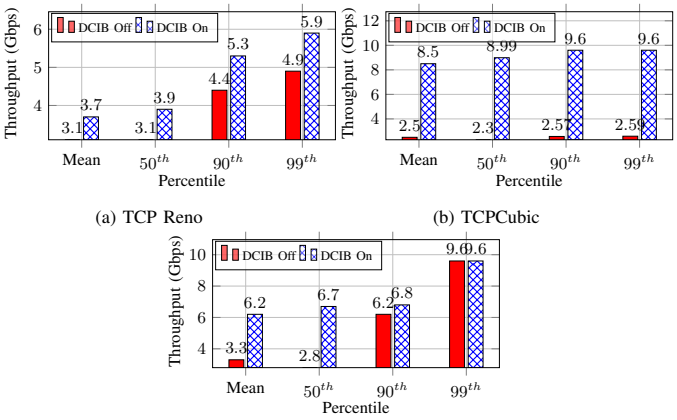
Data: Packet p
if $p.ECN == 0$ **then**
    /* Packet marked for drop */
    Generate Router-to-Router ACK ;
    Send Ack to Router A;
end
Transmit packet p;

Figure 9. Packet processing at Q_{rcv}



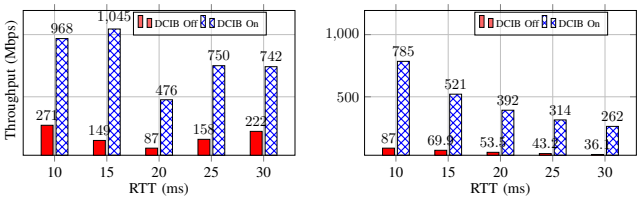
(a) Simulation Topology
(b) DCI throughput (Reno)
(c) DCI throughput (Cubic)
(d) Throughput CDF (Reno)
(e) Throughput CDF (Cubic)

Figure 10. DCIB increases the average throughput by 4x and 6x when packet drop occurs in all packets or in selected packets only.



(a) TCP Reno
(b) TCPCubic
(c) TCPBBR

Figure 11. With no artificial loss, DCIB increases the average throughput by 24% in average.



(a) Cubic
(b) BBR

Figure 12. 5x and 7x higher throughput compared to TCP Cubic and BBR.

percentile. The main reason is BBR ignores to a certain extent packet loss to achieve high throughput.

C. DCIB with Limited Buffer Routers:

Because some edge router might have limited buffer capacity, we also explored using a low-priority buffer to store the selected packets while transmitting them when the main queue is idle. Hence, the buffer requirements are expected to be lower as long as the average transmission rate is lower than the link capacity. Moreover, we configure the end-to-end RTT to represent different sets of networks with different RTT values starting from $10ms$ up to $30ms$. Figure 12a and 12b demonstrate that DCIB enhances the average throughput of TCP Cubic and TCP BBR by 5x and 7x, respectively.

D. DCIB effect on FCT:

We also repeated the same experiment while generating 1000 flows using the characteristic of web search workload [2]. Flows start time is generated using Poisson distribution to generate an average load equal 80%. The base TCP variant used in this simulation is TCP Cubic. Figure 13 illustrates that DCIB reduces Flow Completion Time (FCT) by up to 88%.

V. Discussion

Performance-Enhancing Proxy (PEP) proposed terminating TCP connections at edge routers to alleviate the issue of DCI inter connectivity [11]. In such an approach, hosts do not need

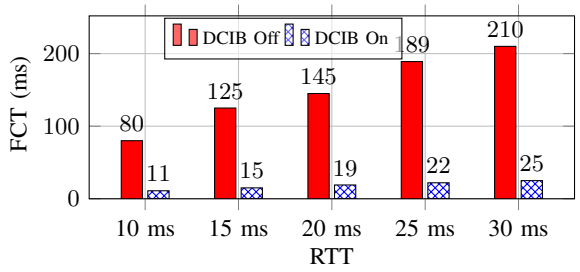


Figure 13. DCIB with stochastic retrans (FCT reduced by 88%)

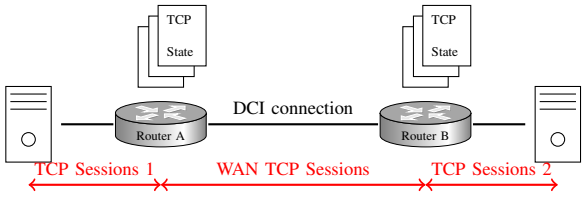


Figure 14. Router A terminates TCP connections and open new connections towards Router B; Router B reestablishes TCP connections towards clients

to wait for the end-to-end ACK to control transmission. However, edge routers need, not only to store all inflight packets but also to keep the state for each connection both to the hosts and to the other edge routers (Figure 14). In addition, managing these connections encounters more delay in processing the whole TCP stack twice (inbound and outbound at the Routers). It would get even more complex for retransmission and buffer management. On the other hand, DCIB does not need to keep any per-connection state, and the buffer requirement is very low and depends on the congestion rate of the communication channel, not the number of connections.

VI. Conclusion

In this paper, we present DCIB to enhance the performance of inter data center communication without changing the TCP protocol at the host. DCIB adds a new method of using ECN marking to selectively drop certain packets instead of sending congestion notification. This allows intermediate routers to drop recommended packets that can easily be recovered at the edge routers without interrupting TCP at the hosts. Our experimental results show that DCIB can increase the DCI throughput by a factor of 6x , 4x, and 7x for both TCP Reno, Cubic, and BBR, respectively while not changing the hosts. In addition, DCIB was able to reduce FCT up to 67%, 85%, and 88% for TCP Reno, Cubic, and BBR, respectively.

Defining an interaction between DCIB and the TCP should allow the hosts to increase their transmission rate even faster by allowing DCIB to select extra packets for drop as probes. Such behavior is left for future work. In addition, intercepting duplicate ACKs at the edge router and retransmitting packets that can be recovered at the edge router is also left for future work.

References

[1] S. Ha, I. Rhee, and L. Xu, "Cubic: A new tcp-friendly high-speed tcp variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, Jul. 2008, ISSN: 0163-5980. DOI: 10.1145/1400097.1400105. [Online]. Available: https://doi.org/10.1145/1400097.1400105.

[2] M. Alizadeh *et al.*, "Data center tcp (dctcp)," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, pp. 63–74, Aug. 2010, ISSN: 0146-4833. DOI: 10.1145/1851275.1851192. [Online]. Available: https://doi.org/10.1145/1851275.1851192.

[3] B. Briscoe, K. D. Schepper, M. Bagnulo, and G. White, *Low Latency, Low Loss, and Scalable Throughput (L4S) Internet Service: Architecture*, RFC 9330, Jan. 2023. DOI: 10.17487/RFC9330. [Online]. Available: https://www.rfc-editor.org/info/rfc9330.

[4] G. Kumar *et al.*, "Swift: Delay is simple and effective for congestion control in the datacenter," in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '20, Virtual Event, USA: Association for Computing Machinery, 2020, pp. 514–528, ISBN: 9781450379557. DOI: 10.1145/3387514.3406591. [Online]. Available: https://doi.org/10.1145/3387514.3406591.

[5] N. Cardwell *et al.*, "Bbrv2: A model-based congestion control performance optimization," in *Proc. IETF 106th Meeting*, 2019, pp. 1–32.

[6] G. Zeng, J. Qiu, Y. Yuan, H. Liu, and K. Chen, "Flashpass: Proactive congestion control for shallow-buffered wan," in *2021 IEEE 29th International Conference on Network Protocols (ICNP)*, 2021, pp. 1–12. DOI: 10.1109/ICNP52444.2021.9651988.

[7] "Network simulations with the ns-3 simulator," 2008, [Online]. Available: https://www.nsnam.org/ (visited on 07/10/2023).

[8] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on networking*, vol. 1, no. 4, pp. 397–413, 1993. DOI: 10.1109/90.251892.

[9] A. Gurtov, T. Henderson, and S. Floyd, *The NewReno Modification to TCP's Fast Recovery Algorithm*, RFC 3782, Apr. 2004. DOI: 10.17487/RFC3782. [Online]. Available: https://www.rfc-editor.org/info/rfc3782.

[10] S. Floyd, D. K. K. Ramakrishnan, and D. L. Black, *The Addition of Explicit Congestion Notification (ECN) to IP*, RFC 3168, Sep. 2001. DOI: 10.17487/RFC3168. [Online]. Available: https://www.rfc-editor.org/info/rfc3168.

[11] J. Griner, J. Border, M. Kojo, Z. D. Shelby, and G. Montenegro, *Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations*, RFC 3135, Jun. 2001. DOI: 10.17487/RFC3135. [Online]. Available: https://www.rfc-editor.org/info/rfc3135.