IFIP Networking 2024 1570991836
2024 IFIP Networking Conference (IFIP Networking)

PB-FS: Postcard-Based Fast Start

Dan Aaronson
Department of Computer Science
Technion
Haifa, Israel
danaa@cs.technion.ac.il

Reuven Cohen
Department of Computer Science
Technion
Haifa, Israel
rcohen@cs.technion.ac.il

***Abstract*—We propose PB-FS (Postcard-Based Fast Start), a rate initialization scheme that uses direct feedback from the switches to quickly correct the rates of new datacenter flows that begin at the line rate and cause congestion. PB-FS is designed to easily integrate into any datacenter congestion control protocol. We evaluate PB-FS in two datacenter environments: a lossless network that runs RoCE, and a lossy network that uses RDMA with selective repeat. We show that PB-FS significantly reduces tail latency of short flows while maintaining throughput, in both lossless and lossy datacenters.**

***Index Terms*—congestion control, datacenter, RDMA**

I. Introduction

Today, datacenter link speeds have reached 400Gbps and they are steadily increasing. These speeds are essential to achieve the low latency and high throughput demands of modern datacenter applications. Fully utilizing these speeds required operators to change fundamental components of the existing datacenter environment, notably adopting RDMA instead of TCP for internal datacenter traffic.

RDMA was originally used over InfiniBand, which is lossless. To run RDMA over Ethernet, which is lossy, datacenter operators had to replace Ethernet with "Converged Ethernet" [1], which is lossless because it uses the priority flow control (PFC) protocol. Running RDMA over converged Ethernet is known as RoCE.

With PFC, switches instruct their neighboring switches to stop sending them packets just before they run out of buffer room. While this prevents packet loss due to congestion, it also reduces the throughput due to head-of-the-line blocking and may create routing deadlocks [1]–[4]. To reduce the negative impact of PFC on the throughput, RoCEv2 runs a congestion control protocol in addition to PFC. A congestion control protocol indeed reduces the negative impact of PFC, but a network running both PFC and congestion control performs worse than the same network running congestion control without PFC.

ISBN 978-3-903176-63-8 © 2024 IFIP

Running RDMA over PFC efficiently is, therefore, still an unsolved problem. Thus, more and more datacenters are running RDMA with congestion control but without PFC. Some change the RDMA protocol to support selective repeat retransmission [5], [6]. This addresses one main issue in the modern datacenter environment, but we believe another main issue has been overlooked, and that is the transition from TCP slow start to immediate transmission at the line rate.

Most congestion control protocols that were designed for lossless RoCEv2 datacenters have abandoned TCP slow start in favor of "fast start", namely, starting the transmission at the line rate [7]–[10]. By doing so, these protocols take advantage of the fact that even if the initial rate is too high, packets will not get lost due to PFC. Fast start allows delay-sensitive flow to complete within one RTT, even if they need to transmit a few tens of packets. However, it also increases congestion, which amplifies the negative effects of PFC discussed above.

The benefit of using fast start has led to using it also in congestion control protocols designed for lossy datacenters [11], [12]. Being lossy, these protocols are susceptible to losing packets during the first RTT of a flow. Even if the retransmission scheme is very efficient, a loss during the first RTT is likely to significantly increase the latency of delay-sensitive flows, from one RTT to two or more RTTs.

Postcard-Based Telemetry (PBT) [13] is a new framework, developed by the IETF. Using PBT, regular packets are marked such that they trigger switches to send special packets, called "postcards", to a predefined recipient. These postcards can contain various per-packet parameters that give insight to the recipient on the current state of the switch. PBT offers a uniquely fast and fine-grained method for relaying congestion information to hosts, making it particularly relevant for congestion control.

In this paper, we propose PB-FS (Postcard-Based Fast Start): a PBT-based rate initialization scheme to quickly correct the rates of new datacenter flows that begin at

978-3-903176-63-8/24/$31.00 ©2024 IEEE

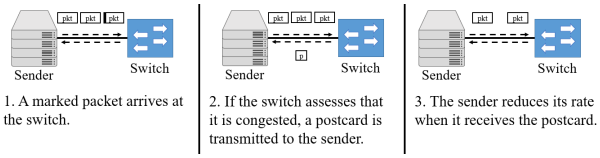


Fig. 1: The PB-FS scheme

the line rate and cause congestion. According to PB-FS, all flows begin at the line rate, but if a flow is "sufficiently large", it is required to mark some packets transmitted during its first RTT for PBT.

As shown in Figure 1, upon receiving a marked packet, switches use a novel algorithm to assess whether they are congested. If they determine that they are, they transmit a postcard to the flow sender to instruct it how to precisely reduce its rate in a way that mitigates congestion but maintains utilization. The proposed scheme completes when the sender receives an acknowledgement that causes the congestion control protocol to update the transmission rate. From this moment on, the congestion control protocol is solely responsible for the sender's transmission rate and it updates the transmission rate relative to the corrected rate attained from PB-FS.

Similar to TCP slow start, PB-FS is designed to easily integrate into any datacenter congestion control protocol. Thus, it does not rely on any specific component of a congestion control protocol for its operation. PB-FS remains constant for any congestion control protocol and its implementation is simple.

The rest of the paper is organized as follows. In §II we discuss the current state-of-the-art of congestion control in high speed datacenters. In §III we give a detailed description of the proposed PB-FS scheme. In §IV we analyze the different parameters of PB-FS and give guidelines for setting their values. In §V we evaluate PB-FS using extensive simulations with ns-3. We show that PB-FS significantly reduces tail latencies for small flows, while maintaining throughput, when it is deployed with DCQCN, TIMELY, and HPCC. In the state-of-the-art lossless environment, the 95th percentile flow completion times (FCTs) are shown to be over eight times shorter for DCQCN and TIMELY, and 10% shorter for HPCC. Finally, §VI concludes the paper.

II. Related Work

The earliest and most well-known rate initialization scheme is TCP Reno's slow start [14]. For many years it was widely accepted as the best approach for achieving high performance and low loss in the internet. CUBIC [15] uses an improved rate initialization scheme, which increases the transmission rate according to a cubic function. This increases the transmission rate faster than TCP slow start for the first few RTTs, then slows down the increase near the slow-start threshold, and thereafter continues increasing the rate rapidly, if there is available bandwidth.

Datacenter link speeds have increased such that a majority of flows can be completed in a single RTT. Thus, starting flows at the line rate (fast start) can greatly reduce latency, if the increased congestion is managed. Congestion control protocols designed for lossless RDMA fabrics that deploy PFC [7]–[10] use fast start. This lossless configuration assists in managing the congestion created by fast start.

Congestion control protocols designed for lossy datacenter networks have been slower to adopt fast start. DCTCP [16] uses TCP Reno's slow start. Amazon's SRD protocol [6] uses a rate initialization scheme similar to CUBIC.

A recent line of work for the lossy datacenter environment studies proactive transport protocols. Proactive transport protocols have receivers or a centralized controller proactively allocate bandwidth to senders as credits, which they then consume to transmit their data. New flows receive credits after the first RTT. Packets transmitted during this "pre-credit phase" are referred to as unscheduled packets, and packets transmitted using credits are referred to as scheduled packets.

In ExpressPass [17], senders do not transmit packets during the pre-credit phase. This hurts short flows that can transmit all their data in a single RTT. In Homa [12] and NDP [11], senders use fast start. This increases congestion, which can lead to loss that increases tail FCTs. To deal with this, NDP creates a lossless network by configuring switches to trim packets to headers when they experience congestion.

III. The PB-FS Scheme

A. Scheme Overview

PB-FS is a rate initialization scheme that starts flows at the line rate and corrects their rates quickly if the network is congested, using feedback messages sent directly from the switches. These feedback messages are called CC-postcards, and they are generated, forwarded and processed while imposing very small overhead. PB-FS can be integrated into every datacenter congestion control protocol, since it does not rely on any specific congestion control component. Further, PB-FS does not require the support of all switches. Thus, it can be implemented gradually, starting with the ToR (Top of Rack) switches, which are more likely to experience congestion. PB-FS consists of a sender's packet marking algorithm, a switch's CC-postcard generation algorithm,

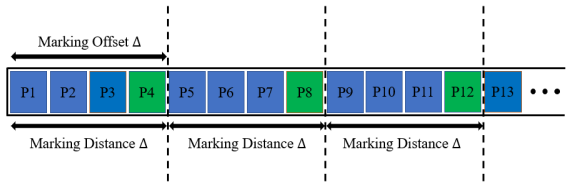


Fig. 2: The sender's packet marking algorithm

and a sender's CC-postcard reaction algorithm. The receiver does not have any role in this scheme.

The sender uses PB-FS only for flows that are "not too short" to react to CC-postcards. The sender's algorithm marks some of the packets transmitted during the first RTT as "subject to CC-postcards." When a switch receives a marked packet, it decides whether or not to send the sender a CC-postcard. CC-postcards are assigned high network priority, so they arrive with minimum latency. When a sender receives a CC-postcard, it reduces its rate, while taking into account the information provided by the switch in this CC-postcard. After receiving the first ACK that updates the sender's transmission rate, the sender stops using PB-FS by ignoring future CC-postcards, and continues to follow the regular congestion control protocol rules. The congestion control protocol updates the transmission rate that might have been affected by the PB-FS scheme. We refer to the period of time between the transmission of the first packet and receiving this ACK as the "PBT Interval."

B. The Sender Packet Marking Algorithm

PB-FS is used only by new flows that require a time period of at least τ to transmit all their data when the sender works at full speed. Thus, the minimum size of a flow that uses PB-FS is $T \cdot \tau$ bytes, where T is the sender transmission rate. The value of τ is determined such that flows that are likely to finish transmitting all their data before a CC-postcard arrives will not receive a CC-postcard. In §IV we discuss the value of τ.

Figure 2 demonstrates the sender's packet marking algorithm. The sender marks every Δ packets, starting from the Δ'th packet, as "PBT-triggering packets." Only these packets can trigger the transmission of a CC-postcard by a switch, if they are received while the switch is congested. The sender stops marking packets when it receives the first ACK or the first CC-postcard. Any CC-postcards generated due to the PBT-triggering packets transmitted after this time are either ignored or have marginal impact on the sender's rate. In [13], there are several suggestions how to mark packets as PBT-triggering. In our simulations (§V), we use a single bit from the TOS field of the IPv4 header.

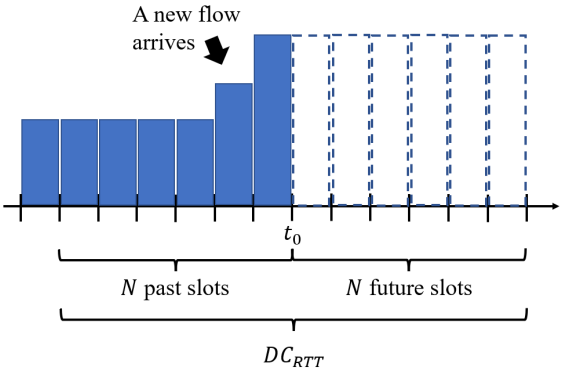


Fig. 3: The congestion assessment

Δ is an important parameter of PB-FS, which sets the marking offset and the marking distance. The role of the marking offset is to delay the arrival of the first PBT-triggering packet to a switch until the switch has received enough packets from the considered flow to determine the impact of this flow on its relevant outgoing buffer. The marking offset, however, should not be too large, because we want the sender to react as early as possible to potential congestion.

The role of the marking distance is to limit the number of CC-postcards generated for each flow. By increasing the length of the marking distance, we reduce the number of PBT-triggering packets and, therefore, reduce the maximum number of CC-postcards sent to the sender. In §IV we discuss the value of Δ.

C. The Switch CC-Postcard Generation Algorithm

A switch that receives a PBT-triggering packet and determines that the relevant output port is congested sends a CC-postcard to the sender. To ascertain whether a given output port is indeed congested, the switch remembers the recent queue length and incoming rate for each output port. To this end, the switch divides the time axis into equal time slots such that $\mathrm{DC_{RTT}}$ is equal to $2N$ slots. We discuss the value of the parameter N in §IV. For each output port, the switch remembers the total number of bytes it receives and needs to forward to the considered port during each of the last N time slots. Let the number of bytes received during these slots be $I_T^1, I_T^2, ..., I_T^N$, where I_T^N is the number of bytes received during the most recent slot. The switch also remembers the queue length for each output port at the beginning of each slot. Let these lengths be $Q^1, Q^2, ..., Q^N$, respectively.

Suppose that a PBT-triggering packet is received by a switch for a certain output port. Let t_0 be the time this packet is received. The switch decides that this port is congested according to the following definition.

Definition 1. A port is considered congested at t_0 if the total number of bytes predicted to be received during the $2N$ slots in $[t_0 - \frac{\mathrm{DC_{RTT}}}{2}, t_0 + \frac{\mathrm{DC_{RTT}}}{2}]$ exceeds the maximum number of bytes the switch can transmit during $2N$ consecutive time slots.

Figure 3 depicts this idea. As it shows, the switch takes into consideration the past N slots and the future (unknown) N slots. Each future N slot is predicted to have the same input rate as the last slot. As shown in §IV, this provides a good estimate of the future arrival rate when the scheme parameters are appropriately chosen.

The number of bytes received for the considered output port p during the previous N slots $[t_0 - \frac{\mathrm{DC_{RTT}}}{2}, t_0]$ is:

$$I_p^{\mathrm{before}} = \sum_{i=1}^{N} I_T^i.$$

The switch predicts that the number of bytes it will receive during the next N slots $[t_0, t_0 + \frac{\mathrm{DC_{RTT}}}{2}]$ is:

$$I_p^{\mathrm{after}} = N \cdot I_T^N,$$

and the total number of bytes predicted to be received during the $2N$ slots $[t_0 - \frac{\mathrm{DC_{RTT}}}{2}, t_0 + \frac{\mathrm{DC_{RTT}}}{2}]$ is:

$$I_p = I_p^{\mathrm{before}} + I_p^{\mathrm{after}}.$$

The maximum number of these I_p bytes that can be transmitted by the relevant output port during these $2N$ slots is:

$$T \cdot \mathrm{DC_{RTT}} - Q^1, \tag{1}$$

where T is the port's transmission rate and Q^1 is the number of bytes in the buffer of port p at $t_0 - \frac{\mathrm{DC_{RTT}}}{2}$. These bytes must be transmitted before the data received after $t_0 - \frac{\mathrm{DC_{RTT}}}{2}$. If the total number of bytes predicted to arrive for the considered port is greater than $T \cdot \mathrm{DC_{RTT}} - Q^1$, a CC-postcard will be sent. The CC-postcard contains the switch ID (e.g., MAC address), the flow ID (e.g., some fields from the packet headers), the value of I_p, the value of Q^1 (recall that the switch remembers the queue length for each of the last N slots), the port transmission rate (T), and the number of bytes received recently by this port. The latter bytes allow the sender to calculate the required rate reduction as we shall see in §III-D.

D. The Sender Response to CC-Postcards

A sender that uses PB-FS for a certain flow can receive for this flow several CC-postcards from the same or different switches. We now show how each CC-postcard allows the sender to estimate the available bandwidth of the relevant port of the sending switch, and calculate its rate reduction accordingly. Recall that

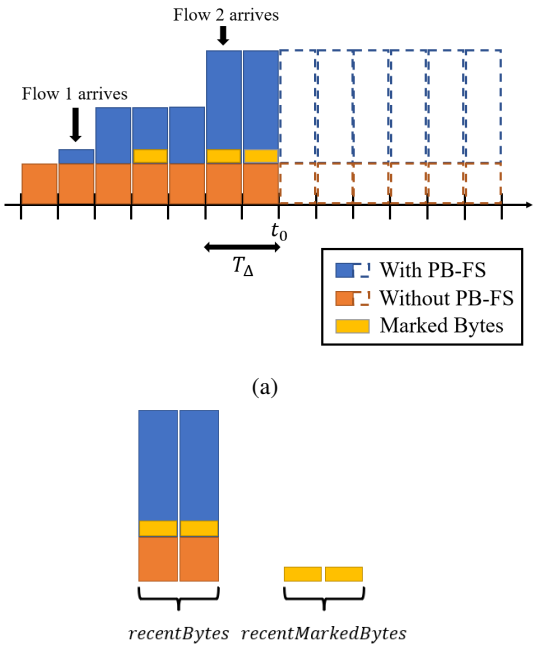


Fig. 4: Traffic distribution estimation

if a sender receives a CC-postcard after its PBT Interval expires, this CC-postcard is ignored.

Let t be the time the CC-postcard is received by the sender. The sender reduces the rate of the flow under consideration as if the CC-postcard is detailing the switch's expected state in the following $\mathrm{DC_{RTT}}$ $[t, t + \mathrm{DC_{RTT}}]$. In reality, the CC-postcard details the switch's state during a $\mathrm{DC_{RTT}}$ time interval before time t. For example, if the CC-postcard takes $\frac{\mathrm{DC_{RTT}}}{2}$ to arrive at the sender, then it details the time interval $[t - \mathrm{DC_{RTT}}, t]$ because the switch predicted $\frac{\mathrm{DC_{RTT}}}{2}$ into the future when the CC-postcard was sent. We tolerate this inaccuracy caused by the unavoidable delay of network feedback.

To decide how much to reduce the rate of the relevant flow, the sender must take into account both flows that use PB-FS and flows that do not use it. Recall that flows do not use PB-FS if they are "too short" (shorter than $T \cdot \tau$ bytes) or if they have completed their PBT Interval. Some I_p bytes that the switch is predicted to receive during the $\mathrm{DC_{RTT}}$ time interval related to the relevant CC-postcard are transmitted by such flows. Let I_{nonPBT} be the number of these bytes and define:

$$I_{PBT} = I_p - I_{nonPBT}.$$

The sender predicts that the number of bytes transmitted by flows that do not use PBT during the following $\mathrm{DC_{RTT}}$ will remain the same as the number of bytes

detailed by the CC-postcard. Thus, it must consider the value of I_{nonPBT} when it calculates the available bandwidth of the sending switch.

The values of I_{PBT} and I_{nonPBT} are estimated by the sender using the information carried by the CC-postcard regarding the number of bytes recently received for the considered port. The number forwarded by the CC-postcard relates to the T_Δ interval before the CC-postcard was sent. T_Δ is the time it takes the sender to transmit Δ full sized packets (recall that Δ is the marking distance).

Figure 4 demonstrates this for the case where T_Δ is equal to two time slots. Let t_0 be the time a PBT-triggering packet is received. In Figure 4a, we show the distribution of the received bytes between flows that use PB-FS and flows that do not use it during the interval $[t_0 - \frac{\mathrm{DC_{RTT}}}{2}, t_0 + \frac{\mathrm{DC_{RTT}}}{2}]$. I_{PBT} is the sum of the bytes colored blue and yellow and I_{nonPBT} is the sum of the bytes colored orange. For the estimation, the switch includes the number of bytes received during the denoted T_Δ interval as shown in Figure 4b. $recentBytes$ is the number of bytes received from all packets, and $recentMarkedBytes$ is the number of bytes received only from PBT-triggering packets. Consider the ratio of $recentMarkedBytes$ to $recentBytes$. We observe that by multiplying this ratio by Δ, we get an estimate of the percentage of I_p originating from flows that use PB-FS, because each marked packet indicates the arrival of $\Delta - 1$ additional packets. Thus, the number of bytes received during $[t_0 - \frac{\mathrm{DC_{RTT}}}{2}, t_0 + \frac{\mathrm{DC_{RTT}}}{2}]$ that were transmitted by flows that use PB-FS is:

$$I_{PBT} = \Delta \cdot \frac{recentMarkedBytes}{recentBytes} \cdot I_p$$

The sender then calculates that the bandwidth available at the considered output port for flows that use PB-FS is:

$$I = \max\{T \cdot \mathrm{DC_{RTT}} - Q^1 - I_{nonPBT}, I_{min}\},$$

which follows from Eqs. 1 and 2. Here, I_{min} is a lower bound on the value of I, which guarantees that flows that use PB-FS will always receive some portion of the bandwidth. For our simulations, we set I_{min} to 20% of the network's maximum BDP. Lastly, the sender calculates that the multiplicative rate reduction factor required by all flows that use PB-FS and traverse the considered switch in order for the switch port to receive fewer bytes than this port can transmit is:

$$c = \frac{I_{PBT}}{\mu \cdot I},$$

where μ is a constant close to 1 (e.g., 0.95) representing the target utilization. The multiplicative rate reduction is always relative to the sender's maximum rate, so the proposed new rate is:

$$NewRate = \frac{MaxRate}{c}.$$

After calculating the new rate, the sender must also consider the rates proposed by the most recent CC-postcards received from every other switch. It updates its transmission rate to be the minimum of these rates.

Algorithm 1 formalizes the sender's CC-postcard reaction algorithm. Procedure HandleCCPostcard is executed whenever a CC-postcard is received and Procedure HandleAck is executed whenever an ACK is received. Lines 4–8 are the estimation of the available bandwidth and the calculation of the proposed rate reduction. In line 12, Procedure UpdateRate is called to update the sender's rate.

Algorithm 1 Sender's reaction to CC-postcards. L is an array that contains the most recent value of c calculated from the CC-postcards received from each switch. ID is the identity of the switch that sent the CC-postcard that determined the current rate. It is initialized to nil.

```
1:  procedure HANDLECCPOSTCARD(p)
2:      if ignoreCCPostcards then
3:          return
4:      I_PBT = Δ · p.recentMarkedBytes / p.recentBytes · p.I_p
5:      I_nonPBT = p.I_p − I_PBT
6:      BDP = p.T · DC_RTT
7:      I = max(BDP − p.Q^1 − I_nonPBT, I_min)
8:      c = I_PBT / (μ · I)
9:      L[p.sID] = c
10:     if c > 1 then
11:         NewRate = MaxRate / c
12:         UpdateRate(p, NewRate)
13: procedure UPDATERATE(p, NewRate)
14:     if ID == nil or ID == p.sID or c > L[ID] then
15:         ID = p.sID
16:         CurRate = NewRate
17: procedure HANDLEACK(ack)
18:     if rate was updated then
19:         ignoreCCPostcards = True
```

IV. Analysis

A switch decides that a port is congested according to Definition 1. A possible implementation method is by using the leaky bucket abstraction model [18]. When a packet arrives, a counter is incremented by the packet length. When the counter is greater than 0, its value decreases at a constant rate that reflects the transmission rate. The counter has a threshold C. When this threshold is exceeded, the received traffic

is considered non-conforming. In our scheme, non-conforming traffic triggers the transmission of a CC-postcard to ask the sender to reduce its rate.

The model parameters are the leak rate T and the bucket capacity C. These parameters set the constraints on the received traffic. Assuming that all packets have the same length $\mathrm{PktSize}$, then

$$\mathrm{MBS} = \left\lfloor \frac{C}{\mathrm{PktSize}} \right\rfloor$$

is the maximum burst size and $C = T \cdot \mathrm{DC_{RTT}}$.

As discussed in §III, when a switch receives a PBT-triggering packet, it predicts that the number of bytes it will receive during each of the next N slots is equal to the number of bytes received during the previous slot. Thus, to ensure that this prediction is accurate, a flow must transmit its data during a period longer than $\frac{\mathrm{DC_{RTT}}}{2}$, which is the minimum value of τ.

When PB-FS is used, a flow transmits at the line rate during its PBT interval. Thus, during each of the N slots after the PBT-triggering packet is received, the switch will receive the same number of bytes from this flow. For the prediction to be accurate, the switch must also receive this number of bytes during the previous slot. We conclude that the marking offset Δ must be set such that the transmission time of Δ packets at the line rate is greater than the length of a slot; namely,

$$T_\Delta = \frac{PktSize \cdot \Delta}{T} \geq \frac{\mathrm{DC_{RTT}}}{2N},$$

where T is the sender transmission rate.

From this constraint it follows that by increasing N we can reduce Δ, implying that the first PBT-triggering packet can be transmitted earlier. On the other hand, by reducing N, we reduce the impact of bursts on the prediction accuracy. If a transient burst is received during the last slot, it is falsely predicted to repeat during the next N slots. Hence, by reducing N, the number of bytes received in a given burst is added to a smaller number of future slots.

There are additional considerations for the selection of τ. Consider a flow for which the distance between the sender and the receiver is the maximum, and consider a CC-postcard sent by the switch that is farthest from the sender. The PBT-triggering packet makes one hop less than the maximum number of hops until a CC-postcard is transmitted by the receiver ToR switch. Then the CC-postcard makes a similar number of hops until it is received by the sender. Thus, it takes slightly less than $\mathrm{DC_{RTT}}$ for the CC-postcard to be received by the sender. We refer to this as the maximum CC-postcard RTT. For example, in the topology considered in §V, the maximum CC-postcard RTT is $\frac{5 \cdot \mathrm{DC_{RTT}}}{6}$.

We conclude that flows for which the distance between the sender and the receiver is the maximum and that need less than the maximum CC-postcard RTT to be completely transmitted are not affected by a CC-postcard sent by the receiver's ToR switch, which is the switch most likely to experience congestion. If τ is set to the maximum CC-postcard RTT, all the flows using PB-FS are able to receive a CC-postcard from all the switches they traverse. There are, however, flows that are completely transmitted during less than the maximum CC-postcard RTT, which can benefit from PB-FS; for example, short flows for which the sender and the receiver are connected to the same ToR switch. CC-postcards can also be sent by switches that are closer to the sender. To maximize the number of flows that can benefit from PB-FS, we suggest setting τ to $\frac{\mathrm{DC_{RTT}}}{2}$.

There are three main considerations for the value of Δ. First, the maximum number of CC-postcards that can be sent to a sender by each switch is $\frac{\mathrm{PBTInterval}}{\Delta}$ (the value of the PBT Interval is discussed in §III). Thus, increasing Δ reduces the number of CC-postcards sent. Second, a smaller value of Δ reduces the interval between the time a congestion occurs and the time a CC-postcard is sent. Third, recall that the sender uses the number of bytes received during the T_Δ interval before the CC-postcard was sent to extrapolate the number of bytes received during the previous $\mathrm{DC_{RTT}}$ from flows that use PB-FS and flows that do not use it. Similar to the discussion for the value of N, if a transient burst is received during the T_Δ interval before the CC-postcard was sent, it will damage the accuracy of this extrapolation. Increasing Δ reduces the relative size of the burst compared to the number of bytes received during the T_Δ interval, which reduces its impact on the accuracy of the extrapolation.

Following our simulations, we recommend setting N such that $\frac{\mathrm{DC_{RTT}}}{2N}$ is between the time it takes to transmit four and five full-sized packets. Then, T_Δ is set to be slightly larger than $\frac{\mathrm{DC_{RTT}}}{2N}$ by setting Δ to approximately 6. This provides a good balance for the various considerations discussed above.

V. Evaluation

A. Simulation Configuration

We evaluated PB-FS using extensive ns-3 simulations. To this end, we added PB-FS and IRN [5] to the ns-3 simulator used in [7], which implements several state-of-the-art RDMA congestion control protocols. We then configured the simulator in the following way.

We evaluated PB-FS with a non-blocking topology derived from [7]. The non-blocking topology is a three-

tier FatTree [19], with 320 hosts, 20 ToR switches, 20 Aggregation switches, and 16 Core switches. Each ToR switch is connected to 16 hosts and 4 Aggregation switches. To speed up the simulation time, the speed of each link is configured to be four times slower than [7]. Each host is connected to its ToR switch via a 25Gbps link, while the links connecting switches are of 100Gbps. Each link has a propagation delay of 2μs. In this setup, the $\mathrm{DC}_{\mathrm{RTT}}$ for a standard 1KB RDMA packet is approximately 25μs. The switch buffers were correspondingly shortened from 32MB to 8MB and are shared by all ports.

We evaluated PB-FS in two datacenter environments. The first is a lossless network over which RoCE is executed. To ensure that the network is lossless, the PFC protocol is used. The dynamic threshold above which a port sends a PFC PAUSE message is proportional to this port's transmission rate: a 100Gbps port sends a PAUSE after it uses more than 33% of the switch's free buffer, whereas a 25Gbps port sends a PAUSE (to a host) after it uses more than 11% of the switch's free buffer.

The second environment uses the IRN protocol [5] running over a lossy datacenter, which does not use PFC. IRN differs from the standard RoCEv2 in two aspects: the sender is configured with a sending window that limits the number of outstanding packets and a selective repeat loss recovery mechanism is implemented. The IRN parameters are chosen according to [5]: $N = 3$, $RTO_{low} = 100\mu s$, and $RTO_{high} = 3ms$. In both environments, the network priority of a CC-postcard is higher than that of a data packet and each port is configured to run priority queueing.

The traffic created by the hosts is derived from a real datacenter flow size distribution [20]. According to this distribution, most flows are very short and can be completely transmitted before the senders can receive any feedback from the sender ToR switch. Only around 10% of the flows are sufficiently large to use PB-FS, but they send most of the data and their behavior impacts the performance of the short flows. Traffic is generated such that the average load imposed on each link is approximately 70%, implying that each host transmits data for approximately 70% of the simulation duration. To evaluate the impact of incast, 100 senders are randomly chosen to transmit an equally sized payload to a single receiver. Both short ($\sim 3KB$) and large ($\sim 100KB$) incasts are evaluated.

PB-FS is evaluated with four representative RDMA congestion control protocols: DCQCN [8], TIMELY [9], HPCC [7] and DCTCP [16]. The parameters of PB-FS are set according to §IV. τ is set to $\frac{\mathrm{DC}_{\mathrm{RTT}}}{2}$. N is set to 8, such that the length of a slot, $\frac{\mathrm{DC}_{\mathrm{RTT}}}{2N}$, is approximately the time it takes to transmit 4.5 full-sized packets ($\sim 1,560$ns). Then, Δ is set to 6, such that T_{Δ} is equal to $\sim 2,035$ns.

The considered performance metrics are: median FCT, 95th percentile FCT, 99th percentile FCT, maximum FCT, average throughput, and the number of generated CC-postcards. The maximum FCT is considered for incast traffic, because incast applications, such as Barrier and All-Reduce, must wait for the completion of all flows before they continue running. We also track the PFC pause duration in the lossless environment and the number of dropped packets in the IRN environment.

B. Lossless Environment

We start by summarizing our main observations for the performance of PB-FS in a lossless RoCE environment. Figure 5 shows the 95th percentile FCTs of the short flows with and without PB-FS, using DCQCN, TIMELY, and HPCC[1]. These flows represent $\sim 80\%$ of all the flows in the evaluated trace.

It is evident that both DCQCN and TIMELY significantly benefit from PB-FS, as their 95th percentile FCTs are over eight times shorter. HPCC, which manages to maintain almost empty queues, also benefits from PB-FS, but its improvement is smaller: 10% for flows shorter than 1,000 bytes (50% of all flows).

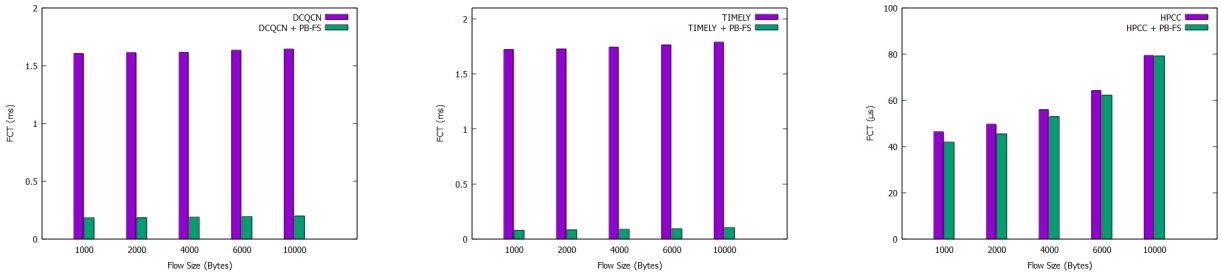
All the above improvements are attributed to PB-FS' ability to significantly reduce the average queue lengths and maximum queue lengths. The maximum queue lengths for DCQCN, TIMELY, and HPCC are reduced by 63%, 87%, and 30%, respectively.

Figure 6a shows the average number of postcards generated for each flow that is large enough to use PB-FS, for DCQCN, TIMELY, and HPCC. We can see that the number of postcards used for each flow is less than four, which we believe is a reasonable overhead. Figure 6b shows the average length of the path traversed by the postcards for HPCC. As expected, most of the congestion occurs in the downstream ports of the destination ToR switches.

C. Lossy Environment

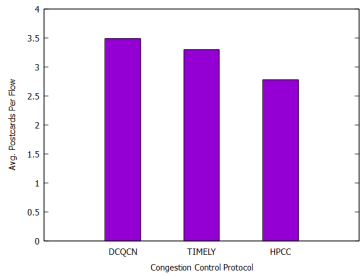
The many known drawbacks of PFC [1]–[4], and in particular the bandwidth loss due to head-of-the-line-blocking, lead us to examine PB-FS in an environment where PFC is disabled. Without PFC, losses can occur even if CC is used. In particular, if flows use fast start, losses are likely to take place during the first RTT, before CC indication is received by the sender. We

[1]DCTCP is not considered because it was designed for a lossy network.

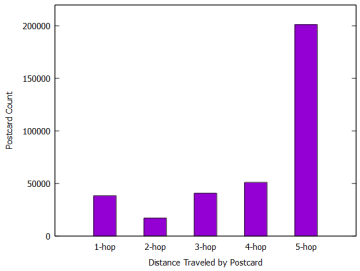


(a) DCQCN (b) TIMELY (c) HPCC

Fig. 5: The 95th percentile FCTs of DCQCN, TIMELY, and HPCC for short flows in the RoCE evaluation, when the average load is ∼70%



(a) Average number of received postcards for DCQCN, TIMELY, and HPCC



(b) Transmitted CC-postcard by distance from switch to source for HPCC

Fig. 6: CC-postcard transmission analysis in RoCE

compare the performance of PB-FS to fast start (FS) and slow start (SS). In FS, the sender starts at the maximum rate, while in SS it starts at the minimum rate, and increases its rate exponentially (as TCP does). DCTCP is used as a standard congestion control protocol for a lossy network. The loss recovery scheme is the standard RDMA RoCEv2, namely, Go-Back-N, and the timeout until a packet is considered lost is set to $100\mu s$. We examine 8MB and 2MB shared switch buffers. With 8MB, no packet loss is observed in the simulation, even when FS is used. Thus, FS performs better than SS. We now discuss the results for 2MB buffers. The remaining configuration is identical to the previous simulations.

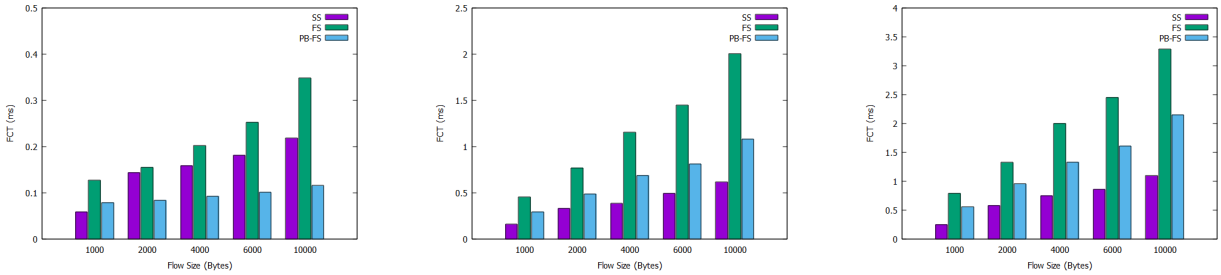
Figure 7 shows the median, 95th percentile, and 99th percentile FCTs for short flows using SS, FS, and PB-FS. For the median FCT, PB-FS's results are at least 70% better than those of SS and FS for all short flows except for one-packet flows. One-packet flows benefit from the smaller queues obtained by SS and they are not affected by the initial rate. For the 95th and 99th percentile FCTs, PB-FS is better than FS, but is worse than SS. This is attributed to the loss percentages for SS, FS, and PB-FS, which are 2%, 15.7%, and 7.9%, respectively. This implies that PB-FS prevents some of the FS losses, but the losses that it cannot prevent increase its tail FCTs.

From this experiment we conclude that PB-FS is not enough to allow disabling PFC. This leads us to evaluate PB-FS with IRN [5], which uses an efficient loss recovery mechanism, based on selective repeat. Figure 8 shows the 95th FCTs of the short flows, with and without PB-FS, using DCQCN, TIMELY, and HPCC. Compared to the RoCE evaluation, the improvement is smaller but still significant. PB-FS improves the 95th percentile FCTs for one-packet flows by 36%, 15%, and 10% for DCQCN, TIMELY, and HPCC, respectively.

IRN achieves smaller average and maximum queue lengths than RoCE and, when PB-FS is deployed, they are reduced even more. PB-FS reduces the maximum queue lengths by 44%, 7%, and 21% for DCQCN, TIMELY, and HPCC, respectively.

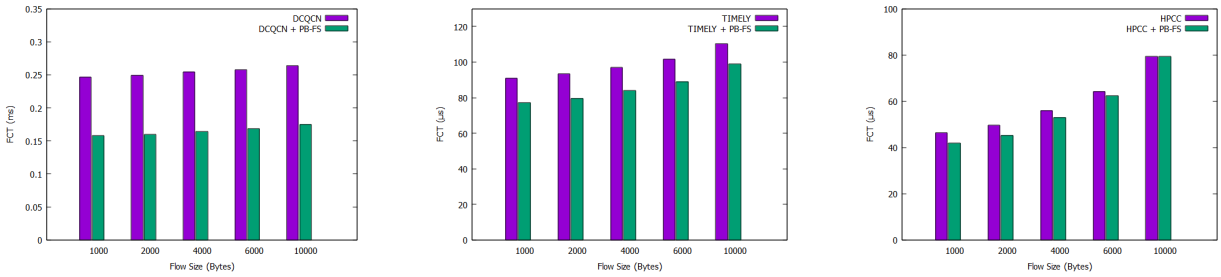
VI. Conclusion

Deploying PFC in datacenters networks to allow running RDMA has introduced several issues. There is significant work to remove PFC by using advanced congestion control schemes and by improving RDMA's loss recovery. We show that the transition to fast start, either with or without PFC, has also negatively impacted performance. We propose PB-FS, a new rate initialization scheme, which balances between gradual



(a) Median FCT (b) 95th percentile FCT (c) 99th percentile FCT

Fig. 7: The median, 95th percentile, and 99th percentile FCT for short flows in a lossy network with DCTCP



(a) DCQCN (b) TIMELY (c) HPCC

Fig. 8: The 95th percentile FCTs of DCQCN, TIMELY, and HPCC for short flows in the IRN evaluation, when the average load is ∼70%

rate increase to an aggressive start. The three algorithms of PB-FS work together to provide precise feedback to new flows as quickly as possible. PB-FS significantly reduces congestion and latency in both lossless and lossy datacenters.

REFERENCES

[1] C. Guo, H. Wu, Z. Deng, G. Soni, J. Ye, J. Padhye, and M. Lipshteyn, "Rdma over commodity ethernet at scale," in SIGCOMM 2016.

[2] S. Hu, Y. Zhu, P. Cheng, C. Guo, K. Tan, J. Padhye, and K. Chen, "Deadlocks in datacenter networks: Why do they form, and how to avoid them," in HotNets 2016.

[3] A. Shpiner, E. Zahavi, V. Zdornov, T. Anker, and M. Kadosh, "Unlocking credit loop deadlocks," in HotNets 2016.

[4] S. Hu, Y. Zhu, P. Cheng, C. Guo, K. Tan, J. Padhye, and K. Chen, "Tagger: Practical pfc deadlock prevention in data center networks," in CoNEXT 2017.

[5] R. Mittal, A. Shpiner, A. Panda, E. Zahavi, A. Krishnamurthy, S. Ratnasamy, and S. Shenker, "Revisiting network support for rdma," in SIGCOMM 2018.

[6] L. Shalev, H. Ayoub, N. Bshara, and E. Sabbag, "A cloud-optimized transport protocol for elastic and scalable hpc," *IEEE Micro*, vol. 40, pp. 67–73, 11 2020.

[7] Y. Li, R. Miao, H. H. Liu, Y. Zhuang, F. Feng, L. Tang, Z. Cao, M. Zhang, F. Kelly, M. Alizadeh, and M. Yu, "Hpcc: High precision congestion control," in SIGCOMM 2019.

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

[9] R. Mittal, V. T. Lam, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, and D. Zats, "Timely: Rtt-based congestion control for the datacenter," in SIGCOMM 2015.

[10] P. Taheri, D. Menikkumbura, E. Vanini, S. Fahmy, P. Eugster, and T. Edsall, "Rocc: Robust congestion control for rdma," in CoNEXT 2020.

[11] M. Handley, C. Raiciu, A. Agache, A. Voinescu, A. W. Moore, G. Antichi, and M. Wójcik, "Re-architecting datacenter networks and stacks for low latency and high performance," in SIGCOMM 2017.

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

[13] H. Song, G. Mirsky, T. Zhou, Z. Li, T. Graf, G. Mishra, J. Shin, and K. Lee, "On-Path Telemetry using Packet Marking to Trigger Dedicated OAM Packets," Internet-Draft draft-song-ippm-postcard-based-telemetry-16, Internet Engineering Task Force, June 2023. Work in Progress.

[14] W. Eddy, "Transmission Control Protocol (TCP)." RFC 9293, Aug. 2022.

[15] S. Ha, I. Rhee, and L. Xu, "Cubic: A new tcp-friendly high-speed tcp variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, p. 64–74, 7 2008.

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

[17] I. Cho, K. Jang, and D. Han, "Credit-scheduled delay-bounded congestion control for datacenters," in SIGCOMM 2017.

[18] J. Turner, "New directions in communications (or which way to the information age?)," *IEEE Communications Magazine*, vol. 24, pp. 8–15, 10 1986.

[19] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in SIGCOMM 2008.

[20] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in IMC 2010.