Dilemma of Proactive Congestion Control Protocols

Kexin Liu, Chen Tian*, Xiaoliang Wang, Wanchun Dou, Guihai Chen
Nanjing University, China

ABSTRACT

Reactive congestion control (RCC) protocols have undergone decades of evolution, where senders first send data packets and then back off when congestion occurs. Recently, there has been a surge of interest in proactive congestion control (PCC) that allocates bandwidth before transmission. Despite its potential, we found that there are certain scenarios where PCC may fall short. In this paper, we aim to provide a comprehensive understanding of PCC and motivate further exploration of this area. We conduct case studies and leverage NS3 simulations to compare state-of-the-art PCC with RCCs, delving into the real dilemma of PCC.

CCS CONCEPTS

• **Networks** → **Transport protocols**; *Data center networks.*

KEYWORDS

Proactive Congestion Control, Datacenter

ACM Reference Format:
Kexin Liu, Chen Tian*, Xiaoliang Wang, Wanchun Dou, Guihai Chen. 2023. Dilemma of Proactive Congestion Control Protocols. In *7th Asia-Pacific Workshop on Networking (APNET 2023), June 29–30, 2023, Hong Kong, China.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3600061.3603123

1 INTRODUCTION

Basic Logic of PCC. Reactive congestion protocols (RCC) have undergone decades of evolution, where flows are first transmitted and then back off when congestion is detected [4, 7]. While in recent years, research communities are exploring the potential advantages brought by proactive congestion protocols (PCC), where data transmission is scheduled through bandwidth allocation from receivers [1–3, 6].

A typical pure proactive design such as ExpressPass [1] proceeds as follows. When a new flow arrives, the sender first notifies the receiver and then waits for the token packets for further data transmission. After receiving the notification, the receiver transmits token packets to schedule the data transmission at a per-token granularity. Tokens compete for bandwidth in networks instead of data packets. And only the tokens that arrive at the sender can trigger the transmission of the corresponding data packets. In this way, the bandwidth used by senders is totally pre-allocated hence the congestion can be eliminated.

PCC is not the ultimate solution. While PCC reduces the possibility of network congestion and provides a relatively steady network environment by bandwidth pre-allocation, the evolution of PCC faces challenges. Recent works show that some PCCs (*e.g.*, ExpressPass, and Aeolus [1, 3]) fall short in scenarios such as when traffic is composed of tiny flows [3, 5]. We also found that other PCC protocols (*e.g.*, NDP and Homa [2, 6]) are not efficient in providing good performance when in-network congestion exists or switch buffer size shrinks.

Table 1: PCC performance under different scenarios.

	ExpressPass [1]	Aeolus [3]	NDP [2]	Homa [6]
Traffic that is composed of tiny flows	×	×		
In-network congestion exists (*i.e.*, load imbalance, or oversubscribed topology)			×	×
Switch buffer size decreases				×
Incast traffic				×

Our contributions. In this paper, we target understanding the dilemma countered by state-of-the-art PCC to motivate further exploration. Taking NDP and Homa as the study case, we first revisit their core design. Then we conduct NS3 simulations to compare NDP and Homa with other state-of-the-art PCCs (*e.g.*, ExpressPass and Aeolus) and RCCs (*e.g.*, DCQCN and HPCC). We investigate the cause of their unsatisfactory performance in these scenarios.

2 CASE STUDY

In this section, we deep dive into the dilemma encountered by proactive congestion control (PCC) protocols. Table 1 summarizes the scenarios where some PCC protocols fall short. It is worth noticing that these scenarios are not uncommon in datacenter networks. ExpressPass and Aeolus do not perform well when traffic is composed of tiny flows. NDP and Homa both can not handle in-network congestion. Besides, Homa encounters a large packet loss ratio when the switch buffer is reduced or under bursty traffic such as incast. Since ExpressPass and Aeolus are already discussed in previous works [3, 5], we mainly analyze NDP and Homa case by case.

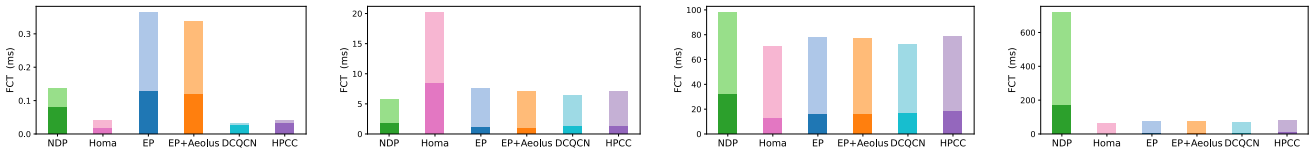
Evaluation settings. We use a leaf-spine topology with a 4:1 oversubscription ratio. It contains 4 core switches, 10 ToRs, and 160 hosts (similar to that in [6]). Each ToR connects to hosts/cores via 100 Gbps links. The buffer size of switches is set to 32M by default and the shared buffer model is used (except for NDP since it sets a small drop threshold on the data queue to avoid a large queuing delay). We use four types of realistic traffic widely used in previous works, covering a wide range of flow sizes [3, 6]. The flow generation follows a Poisson arrival process with a load of 0.8.

2.1 NDP

The logic of NDP. The core design point of NDP is the deterministic drop, *i.e.*, when the queue length exceeds a small threshold, the packet payload is trimmed, and the header is sent to notify the packet loss. It avoids timeout retransmission when packet loss occurs. NDP assumes that congestion only occurs at the last hop of

APNET 2023, June 29–30, 2023, Hong Kong, China

ACM ISBN 979-8-4007-0782-7/23/06.
https://doi.org/10.1145/3600061.3603123



(a) Memcached. (b) Web Server. (c) Cache Follower. (d) Web Search.

Figure 1: Performance comparison under an oversubscribed topology. The deep/light color for each bar represents the average/99th-tail value, respectively.

the network. It only focuses on non-blocking topology and meanwhile uses per-packet source routing to avoid congestion caused by load imbalance. Hence, data packets are simply pulled based on the link rate of the receiver.

NDP can not handle in-network congestion. The mechanism in NDP is effective against congestion at the last hop of the network, but it does not address congestion at the network core. In scenarios where persistent congestion exists at the core switches, *e.g.*, under oversubscribed topology or load imbalance occurs, scheduled data packets could overwhelm the bottleneck. Considering an oversubscribed topology where queue length increases at the fan-in point, NDP actively trims the payload of data packets when the queue length exceeds the preset threshold. The receiver then sends back PULL packets, intending to receive packets at its link rate. All receivers send PULL packets distributedly, without coordination. Given that the topology is oversubscribed, when there are several receivers transmitting PULL packets simultaneously, data triggered by PULL packets could build up at the fan-in point again. It exaggerates in-network congestion. Meanwhile, a relatively small packet-trimming threshold induces a massive packet loss further, which wastes the bandwidth and prolongs the FCTs of flows. When the average flow size is relatively large, the pulling phase dominates and could result in severe congestion. As shown in Figure 1, under Cache Follower and Web Search workloads which are composed of relatively large flows, NDP crashes. The FCTs of NDP are extremely large, *i.e.*, several orders of magnitude than related works.

Per-packet load balancing worth reconsidering. NDP uses per-packet source routing to spray the packets to reduce in-network congestion. It makes the packet pulling more like a scheduling mechanism than a bandwidth allocation since a PULL packet and its corresponding data could pass through a different path. In addition, practical concerns are that it can result in packet reordering which challenges the design of hosts' NICs. It requests NICs to provide a large buffer to handle re-ordered packets.

2.2 Homa

The logic of Homa. In Homa, packets are assigned with different priorities according to the remaining size of their belonging flows. Packets with a high priority are scheduled first at sender hosts and then queued in a high-priority queue in switches. When the receiver receives data packets, it constrains the number of active flows. Tokens are sent to active flows based on a data-driven behavior. With the help of prioritizing scheduling, Homa achieves great performance for tiny flows. It mitigates the Head-of-line (HOL) blocking caused by queuing to an extent, hence small flows can complete transmission quickly. Besides, Homa does not handle congestion in the network core. It uses per-packet spraying to evenly distribute data packets, having the same problem as NDP (§ 2.1).

Prioritization does not intrinsically solve congestion. Priority scheduling is orthogonal to congestion control itself, which could not intrinsically deal with congestion. Since the buffer size of switches is limited, without control over congestion, the queue length can continue to grow when new flows arrive, *i.e.*, when persistent high load or incast traffic pattern exists. Hence, packet loss and timeout retransmission can not be avoided. As shown in Figure 1(b), Homa does not perform well in Web Server workload, for that Homa encounters a large amount of packet loss, *i.e.*, the packet loss rate is around 3% even with a 32 MB shared switch buffer. When the switch buffer decreases, the performance of flows could downgrade further.

Besides, in-network priority queues are not infinite, and in practice, an application can not monopolize all priority queues of switches. When the number of queues is restricted to a smaller value, it could be possible that small flows queue behind large flows when the buffer occupancy increases. Thus HOL-blocking could not be avoided.

Homa does not react to congestion. In Homa, tokens are transmitted according to the arrival rate of data packets, with the overcommitment mechanism to keep only several flows active simultaneously. Besides this mechanism, no additional control for congestion is applied. The number of active flows is static, and could not respond to the variable congestion status of the network. From this perspective, Homa is more like an end-host scheduling or matching protocol rather than a congestion control protocol.

REFERENCES

[1] Inho Cho, Keon Jang, and Dongsu Han. 2017. Credit-scheduled delay-bounded congestion control for datacenters. In *ACM SIGCOMM*.
[2] Mark Handley, Costin Raiciu, Alexandru Agache, Andrei Voinescu, Andrew W Moore, Gianni Antichi, and Marcin Wójcik. 2017. Re-architecting datacenter networks and stacks for low latency and high performance. In *ACM SIGCOMM*.
[3] Shuihai Hu, Wei Bai, Gaoxiong Zeng, Zilong Wang, Baochen Qiao, Kai Chen, Kun Tan, and Yi Wang. 2020. Aeolus: A Building Block for Proactive Transport in Datacenters. In *ACM SIGCOMM*. ACM.
[4] Yuliang Li, Rui Miao, Hongqiang Harry Liu, Yan Zhuang, Fei Feng, Lingbo Tang, Zheng Cao, Ming Zhang, Frank Kelly, Mohammad Alizadeh, et al. 2019. HPCC: High precision congestion control. In *ACM SIGCOMM*.
[5] Kexin Liu, Chen Tian, Qingyue Wang, Yanqing Chen, Bingchuan Tian, Wenhao Sun, Ke Meng, Long Yan, Lei Han, Jie Fu, et al. 2022. PayDebt: Reduce Buffer Occupancy Under Bursty Traffic on Large Clusters. *IEEE Transactions on Parallel and Distributed Systems* (2022).
[6] Behnam Montazeri, Yilong Li, Mohammad Alizadeh, and John Ousterhout. 2018. Homa: A receiver-driven low-latency transport protocol using network priorities. In *ACM SIGCOMM*.
[7] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. 2015. Congestion control for large-scale RDMA deployments. In *ACM SIGCOMM*.