Poster: Reverse-Path Congestion Notification: Accelerating the Congestion Control Feedback Loop

Ralf Kundel*, Nehal Baganal Krishna‡, Christoph Gärtner*, Tobias Meuser*, Amr Rizk‡
*Multimedia Communications Lab, Technical University of Darmstadt, Germany
{ralf.kundel, christoph.gaertner, tobias.meuser}@kom.tu-darmstadt.de
‡University of Duisburg-Essen, Germany
{nehal.baganal-krishna, amr.rizk}@uni-due.de

***Abstract*—Congestion control mechanisms in computer networks rely mainly on a feedback loop having a reaction time equal to the flow RTT. Reducing this feedback time helps the sender to react faster to changing network conditions such as congestion. In this work, we propose reverse-path congestion notification on top of programmable networking switches. Our approach can significantly lower the reaction time, such that the congestion control implementation can adapt much faster to changing network conditions. The proposed approach aims to work with current TCP implementations with no required changes to the communication endpoints. Last, we show how the presented approach could be realized by utilizing off-the-shelf programmable switches.**

***Index Terms*—Congestion Control, AQM, bufferbloat, P4**

I. Introduction and Background

Prevalent transport protocols in computer networks, especially the Internet, use congestion control mechanisms to avoid overloading network resources while utilizing and sharing the available network capacity as best as possible. The working mechanism of congestion control in almost every computer network is as follows: If a congestion situation occurs in the network, the packet experiencing this congestion is either dropped or marked by an Explicit Congestion Notification (ECN) bit. Next, one of the transport protocol endpoints, *e.g.*, TCP sender or receiver, detects the missing packet or the ECN-marked packet leading to a reduction of the sending rate. Decreasing the rate counteracts the overload at the bottleneck and the system eventually converges to a stable and fair bandwidth not overloading the bottleneck. A quick reaction to congestion in the network state is crucial, not only to avoid unnecessary high latency caused by bloated buffers but also to prevent high and bursty packet loss.

This working mechanism can be described as a control loop with a reaction time equal to the Round Trip Time (RTT) of the congestion-controlled flow. Hence, it requires at least one RTT from the time point when the congestion signal is created until congestion control takes effect *at the bottleneck*. In fact, the queueing delay at the bottleneck adds to this reaction time as the *congestion signal* is usually queued together with other packets. Obviously, a lower RTT improves the responsiveness of the congestion control mechanism. This RTT sensitivity was modeled as a relation between the RTT and the required buffer capacity in switches [1], [2].

978-1-6654-4131-5/21/$31.00 ©2021 IEEE

In summary, it is highly desirable to decouple the congestion feedback from the bottleneck to the sender from the path to the receiver and back. To that end, Feldmann *et al.* proposed to send `NACK`s from the bottleneck directly back to the sender [3]. This approach can enormously reduce the control loop reaction time, however, the transport protocol and especially the sender implementation must be adapted to understand the newly introduced NACK packets.

In contrast to introducing NACK packets, the approach presented in the following of this work is transparent to the sender and receiver, which means no adaption is needed. Therefore, our approach can be deployed in a single node where congestion is likely to happen, *e.g.*, a Broadband Network Gateway (BNG) providing Internet access to many customers [4].

II. Approach

We assume a scenario consisting of one TCP sender and one TCP receiver connected via a programmable switch where congestion may arise at the egress port towards the receiver (see Figure 1). Congestion may arise on this path, *e.g.*, due to packet contention on this port, and as a consequence, the egress queue in the switch would temporarily fill up.

Our approach denoted reverse-path congestion notification utilizes existing Explicit Congestion Notification (ECN) mechanisms [5] of TCP. The decision of marking a packet with an ECN bit is determined by any Active Queue Management (AQM) algorithm within the switch. This decision can be made either through a classical algorithm such as Random Early Drop (RED) or a modern, stateful algorithm like CoDel, which can be directly realized in programmable switches [6].

In contrast to related approaches, we perform the marking action as a consequence of the AQM decision in the reverse direction if possible. For that, the packet in the forward direction will not be modified or dropped. Instead, a hash value on the packet flow will be computed and installed in the reverse direction of the switch. Now the switch waits for another packet traversing in the reverse direction that matches this hash value and once detected, this packet will be ECN marked. Note that it is required to swap source and destination IP addresses and L4-ports in order to match packets in the reverse action. The basis of this work is the programming language P4 which is the current de-facto standard for programming switches [7].

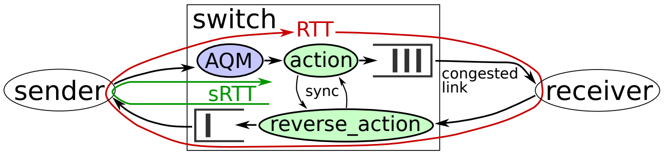

Fig. 1. Design of reverse path congestion notification.

For this approach to work, two assumptions must be fulfilled: 1) The sender and receiver must support TCP ECN signaling. 2) Data flows and their acknowledgment packets traverse the same switch in both directions.

Note that both preconditions can be verified *within the data plane* of the switch for each traversing packet:

(1) Dropping a single TCP ACK packet does not influence the sender rate as the acknowledgment number is cumulative. Therefore, ECN marking in the reverse direction is still valid. ECN-capable flows can be easily detected in the forward direction by checking the corresponding header fields in the IP and TCP header. If a flow is not ECN-capable, the switch can handle it in the traditional way in the forward direction (`fallback #1`).

(2) If a hash value is installed in the reverse direction of the switch and if no ACK-packet was marked within a *timeout* period, future packets should be marked or dropped in the forward direction as a fallback (`fallback #2`).

This data plane behavior is summarized in the pseudocode of Listing 1 and aims at a realization in P4 programmable switches or similar architectures. Note that this algorithm was intentionally kept simple since a realization in programmable switching hardware is intended.

III. Expected Gains of Reverse-Path Congestion Notification

By applying the approach presented above, the feedback time of congestion to the sender can be significantly reduced. Specifically, as shown in Figure 1, the time between congestion occurrence and the reduction of the sending rate is reduced from RTT to $signaling\ RTT$ ($sRTT$). The value $sRTT$ describes the latency from the sender to the congested switch and back. Depending on the position of the switch along the communication path, this reduction could be significant. In addition to the shorter $sRTT$ the connection, the egress queue of the congested switch port is *not* in the control loop. If the queue is filled in case of congestion, this additional queueing latency is not added to the time for signaling congestion to the sender, which is a strong advantage.

```
Packet p; AQM aqm; StatefulRegister reversePath;
//forward direction
if (aqm.notifyCongestion()):
  if (p.isECNenabled()):
    if (!reversePath.hasTimeout(p.flowHash)):
      reverePath.installECNrule(p.flowHash)
    else:
      p.markECN() //fallback #1
  else:
      p.drop() //fallback #2
//reverse direction
if (reversePath.hasECNrule(p.flowHash)):
  p.markECN()
  reversePath.resetECNrule(p.flowHash)
```

Listing 1. Pseudocode of Reverse-Path Congestion Notification that can be realized in P4-programmable data planes.

Given the previously mentioned example of a BNG, providing Internet access to several vDSL-customers of an ISP, we can assume the following numbers to be realistic in downstream direction: $50ms$ queuing delay, $50ms$ latency (both-way) divided into $40ms$ before the bottleneck and $10ms$ after the bottleneck. The proposed approach would lead to a signaling RTT reduction from $100ms$ to an $sRTT$ of $40ms$. In the upstream direction, the RTT can be reduced to the RTT between the local computer and residential gateway, which is an even higher benefit. For that, the proposed reverse-path congestion notification would be installed within the customer's residential gateway instead of the BNG. Further, a potential over-reaction of the congestion control due to dropped and marked packets in parallel must be investigated in future work.

IV. Conclusion and Next Steps

The flow RTT is a crucial factor influencing the effectiveness of congestion control mechanisms in computer networks. Here, we presented the idea of reverse-path congestion notification to reduce the congestion feedback time and, by that, enable a much faster reaction to changing network conditions. The presented approach can be readily deployed using programmable switches and does not require changes to the transport protocol end-points. We estimate a significant reduction of the feedback time, especially for residential upstream scenarios. In future work, we will focus on realizing this approach in programmable switches and residential gateways.

Acknowledgment

This work has been supported by Deutsche Telekom through the Dynamic Networks 9 project, and in parts by the German Research Foundation (DFG) as part of the projects B1, B4, and T3 within the Collaborative Research Center (CRC) 1053 MAKI as well as the project SPINE.

References

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

[2] C. Villamizar and C. Song, "High Performance TCP in ANSNET," *SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 5, p. 45–60, 1994.

[3] A. Feldmann, B. Chandrasekaran, S. Fathalli, and E. N. Weyulu, "P4-enabled network-assisted congestion feedback: A case for nacks," in *Workshop on Buffer Sizing*, ser. ACM BS '19, 2019.

[4] R. Kundel, L. Nobach, J. Blendin, W. Maas, A. Zimber, H.-J. Kolbe, G. Schyguda, V. Gurevich, R. Hark, B. Koldehofe, and R. Steinmetz, "OpenBNG: Central office network functions on programmable data plane hardware," *International Journal of Network Management*, 2021.

[5] S. Floyd, "Tcp and explicit congestion notification," *SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 5, p. 8–23, Oct. 1994.

[6] R. Kundel, A. Rizk, J. Blendin, B. Koldehofe, R. Hark, and R. Steinmetz, "P4-codel: Experiences on programmable data plane hardware," in *IEEE International Conference on Communications (ICC)*, 2021, pp. 1–6.

[7] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, p. 87–95, 2014.