Expanding the Design Space for In-Network Congestion Control on the Internet

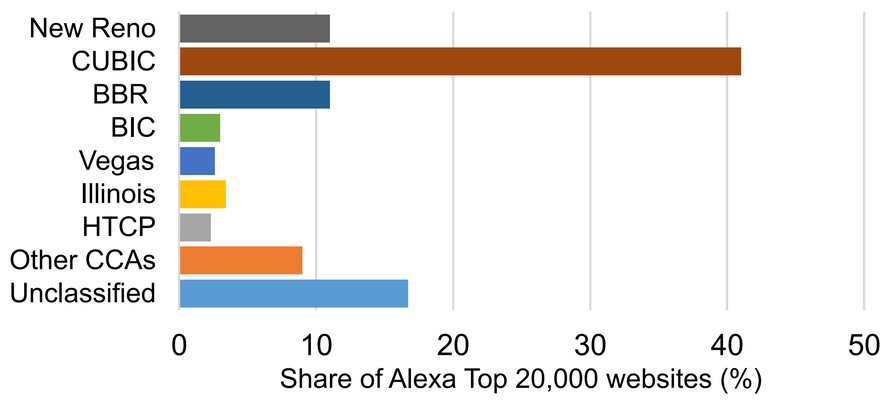
ACM SIGCOMM
Posters and Demos
August 4-8, 2024
Sydney, NSW, Australia

Ayush Mishra[1], Harsh Gondaliya[2], Lingesh Kumaar[1,3], Archit Bhatnagar[1], Raj Joshi[1,4], Ben Leong[1]

[1]National University of Singapore, [2]UC San Diego, [3]BITS Pilani, [4]Harvard University

Motivation

The Internet's Congestion Control Landscape is more heterogeneous than it has ever been before.



Congestion Control Algorithm (CCA) diversity on the Internet according to a measurement study in 2023 [1]

The flow-level fairness implications of different CCAs competing on the Internet is a well known issue.

There is even active deployment of ultra aggressive CCAs like TCP Brutal that speed up instead of slowing down in reponse to congestion signals.

There is a need to curbing degree of unfairness different flows can inflict on each another on the Internet.

[1] Keeping an Eye on Congestion Control in the Wild with Nebby, Mishra et al. Proceedings of SIGCOMM 2024

Design Goals

There is a need for some in-network mechanism that can bound a flow's aggression based on the state of congestion and the volume of traffic at the bottleneck.

Ideally, such a mechanism would have to be

1. **Agnostic to End-hosts**
   It should work with all end hosts, regardless of what CCA they chose to deploy
2. **Able to Enforce send rate reduction**
   Since end hosts have little incentive to maintain fairness, it shoud work even if the end-host ignores all implicit and explicit congestion signals

Moreover, it should also scale well on the Internet.

Case for an rwnd-based solution

All congestion control algorithms limit the number of packets in flight using the minimum of the application set recieve window (rwnd) and the internally congestion window (cwnd).

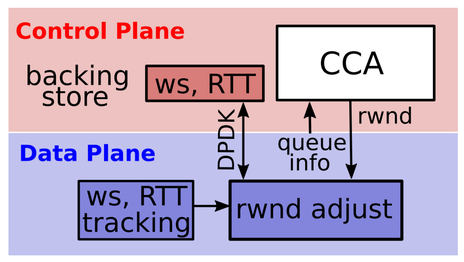
Since the rwnd is exposed in the TCP packet header, it can be utilized as a natural clamping mechanism by any in-network congestion control solution.

As long as the rwnd set by the in-network CCA is lower than the application set rwnd, it will have no impact on the app-level flow control.

Flowtamer Solution Sketch

We present Flowtamer, which is an actuation framework to allow researchers to easily implement rwnd-based in-network congestion control solutions on programmable switches.

Flowtamer provides a high-level API to the control plane. This API provides basic queuing information and allows the control plane to set per-flow rwnd via the dataplane.



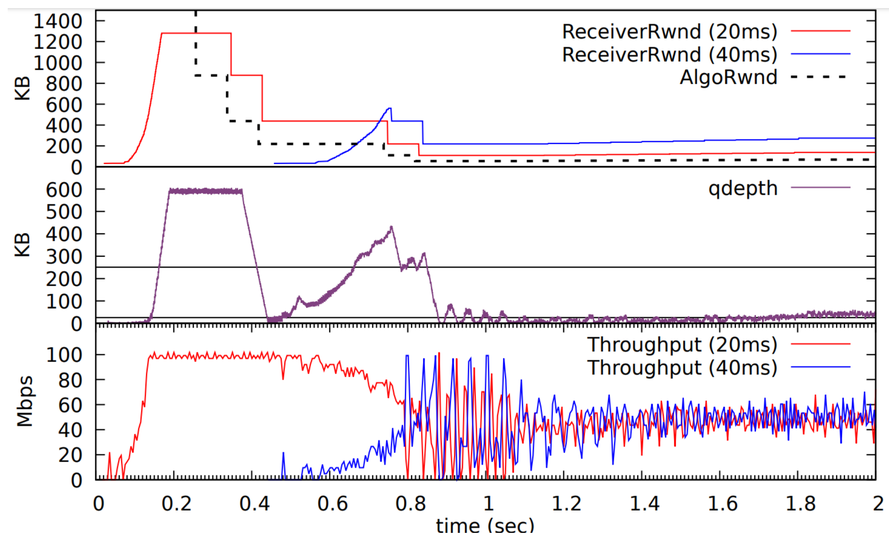
Flowtamer design

The Flowtamer dataplave handles the following challenges that come with setting an rwnd for target flow:

1. **Window Scaling**
   Each flow can have a unique window scaling (ws) factor with which the rwnd value set in the header is multiplied with. Flowtamer automatically tracks this for each flow
2. **TCP Checksum**
   Flowtamer recomputes and modifies the checksum after modifying the rwnd value in the header.
3. **Compensating for different RTTs**
   Flowtamer also scales the rwnd by the per-flow RTT to allow to have same clamping point for all flows.

Proof of Concept

As a proof of concept, we show how a simple AIMD-style rwnd-based algorithm is able to eliminate RTT unfairness between two BBR flows with different RTTs (20ms and 40ms)



Future Work and Limitations

Flowtamer has some natural limitations. It requires path symmetry at the bottleneck and can't work with UDP and QUIC flows that do note expose their rwnd values unencrypted in the header.

In the future, we plan to improve Flowtamer's per-flow state management and include support for flow cardinality estimation.