Need for Speed: Minimizing the Impact of Bufferbloat

Mohit P. Tahiliani

Associate Professor

Department of Computer Science & Engineering

National Institute of Technology Karnataka, Surathkal

tahiliani@nitk.edu.in

{github, gitlab}: mohittahiliani

Outline of the presentation

- Performance requirements of modern Internet systems
  - Bandwidth and Latency
  - Classifying latency requirements
- The problem
  - Bufferbloat: suffering due to extra buffering
  - Reactive Congestion Control in TCP
  - Impact of Bufferbloat on Latency Sensitive Applications
- Potential solution: Active Queue Management
  - AQM algorithms: Goals and Challenges
  - Overview of CoDel and PIE
  - PIE: implementation issue in the Linux kernel (fixed)
  - Overview of FQ-PIE
- Low Latency Low Loss Scalable Throughput (L4S)
  - Scalable congestion control mechanism + AQM algorithm + ECN signaling
  - Key takeaways

Performance requirements of modern Internet systems

Many components of Latency

- Propagation delay

time required for a packet to travel from the sender to receiver, which is a function of distance over speed with which the signal propagates.

- Transmission delay

time required to push all the packet's bits into the link, which is a function of the packet's length and data rate of the link.

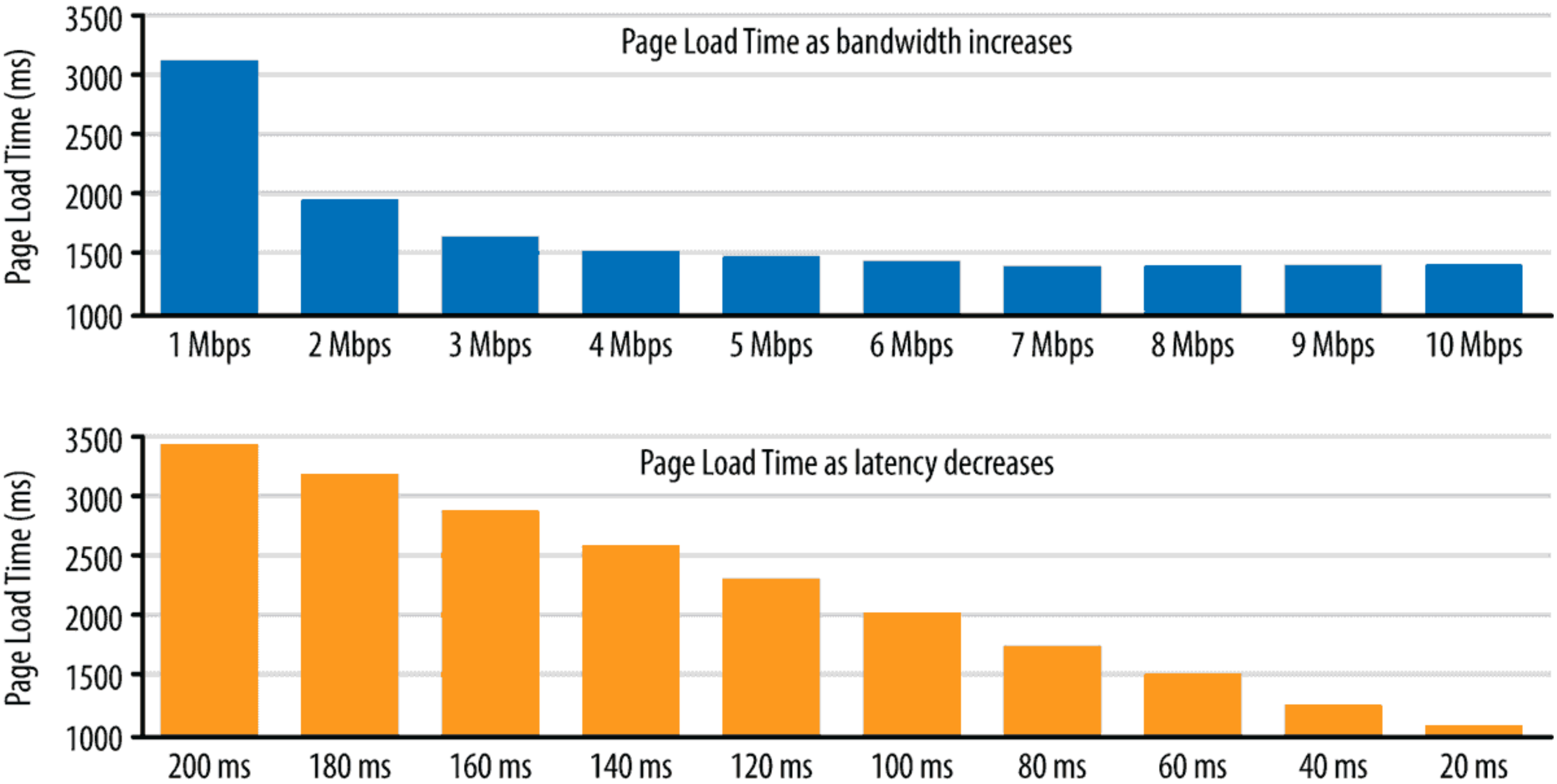
- Processing delay

time required to process the packet header, check for bit-level errors, and determine the packet's destination.

- Queuing delay

time the packet is waiting in the queue until it can be forwarded.

Bandwidth and Latency



Ref.: Ilya Grigorik's book on 'High Performance Browser Networking'

Classifying latency requirements

Three popular categories:

Latency sensitive (or interactive) traffic

Examples: VoIP, audio/video conferencing, online games, remote health monitoring services, IoT based home automation services, and others.

Requirement: minimum latency

Latency tolerant (or elastic, or non-critical) traffic

Examples: File transfers, emails with attachments, image / video sharing applications, backup services, Application / OS updates, and others

Requirement: maximum throughput

Streaming content (or timely) traffic

Examples: Services such as Netflix, Hotstar, Prime Video, and others

Requirement: Less jitter and Consistent throughput

The problem

Bufferbloat: suffering due to extra buffering

- ❑ Memory is inexpensive
- ❑ *Side effect*: Bloated buffers at routers!
- ❑ *Bufferbloat*: large queuing delays

Bufferbloat

Bufferbloat is the undesirable latency that comes from a router or other network equipment buffering too much data.

The Bufferbloat projects provide a webspace for addressing chaotic and laggy network performance. We have a number of projects in flight:

- **What Can I Do About Bufferbloat?** If you have bad latency/lag, or someone has told you there is Bufferbloat in your network, this page lists several steps you can take to measure the bloat in your network, and to solve it completely.
- **The Bufferbloat Project** has largely addressed latency associated with too much buffering in routers. The cake algorithm (and its predecessors CoDel and fq_codel algorithms) are the first fundamental advances in the state of the art of network Active Queue Management in many, many years. These algorithms have been deployed in millions of computers, and reduce the induced delay from competing traffic on a bottleneck link to the order of 20 msec.
- **The Make-Wi-Fi-Fast Project,** with many of the same team members as the Bufferbloat project, intends to improve Wi-Fi's speed and use of the spectrum by inserting CoDel/fq_codel into the Wi-Fi queues, and actively measuring the power required for successful transmission, in order to minimize contention and interference on the RF channel. As of early 2017, our efforts to remove queueing latency and add Airtime Fairness to Wi-Fi stacks have borne fruit. See the Make Wi-Fi Fast Status page.
- **The Request to FCC for Saner Software Policies** is a response to Docket ET 15-170 which appears to require vendors to lock down the software in Wi-Fi routers, prohibiting experimentation and field testing of new techniques. Read the Press Release and our Letter to the FCC

These projects are all united by a desire to:

- Gather together experts to tackle networking queue management and system problem(s), particularly those that affect wireless networks, home gateways, and edge routers
- Spread the word to correct basic assumptions regarding goodput and good buffering on the laptop, home gateway, core routers and servers.
- Produce tools to demonstrate and diagnose the problem
- Make experiments in advanced congestion management
- Produce patches to popular operating systems at the device driver, queuing, and TCP/ip layers to fix the problems.

Recent Updates

Oct 20, 2023 Wiki page
What Can I Do About Bufferbloat?

Dec 3, 2022 Wiki page
Codel Wiki

Jun 11, 2022 Wiki page
More about Bufferbloat

Jun 11, 2022 Wiki page
Tests for Bufferbloat

Dec 7, 2021 Wiki page
Getting SQM Running Right

Find us elsewhere

Bufferbloat Mailing Lists
#bufferbloat on Twitter
Google+ group
Archived Bufferbloat pages from the Wayback Machine

Sponsors

Ref.: https://www.bufferbloat.net/projects/

Reactive Congestion Control in TCP

❑ widely deployed in the Internet (e.g., CUBIC TCP)

❑ sending rate increases until there is a packet loss

❑ packet loss occurs only when the bloated buffers overflow

❑ Result: very high queuing delays!

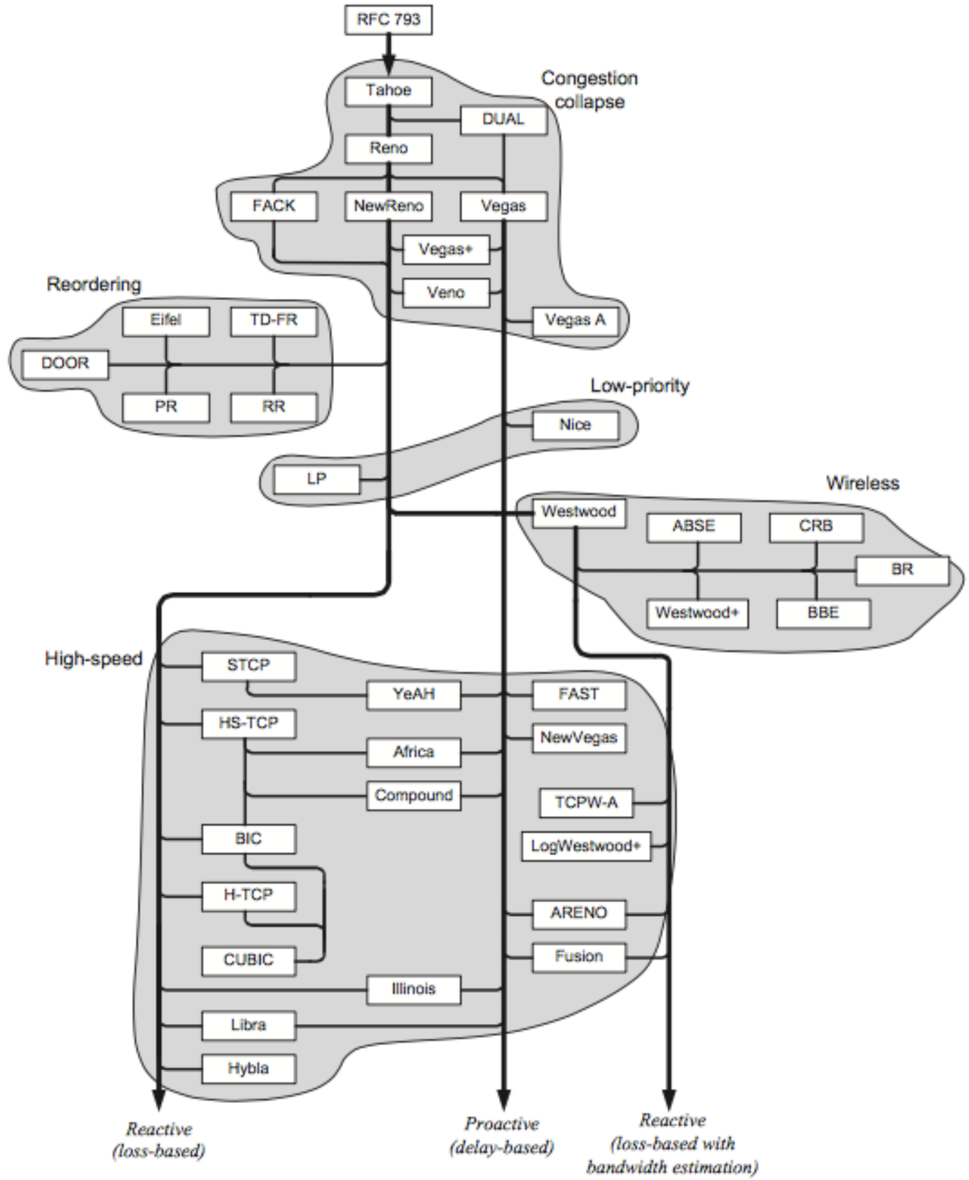


Image Source: Host to host congestion control for TCP, IEEE Communication Surveys and Tutorials, 2010

Impact of Bufferbloat on Latency Sensitive Applications

- High queuing delays and frequent packet losses for latency sensitive traffic

Source: Kennedy, J., Armitage, G., & Thomas, J. (2017). Household bandwidth and the 'need for speed': Evaluating the impact of active queue management for home internet traffic. Australian Journal of Telecommunications and the Digital Economy, 5(2), 113.

Can WMM help solve the problem of Bufferbloat?

- ❑ Probably not!
- ❑ WMM manages traffic, it does not manage buffers.
- ❑ Update traffic priorities and filters frequently (but is it worth?)

What Can I Do About Bufferbloat?

Bufferbloat is high latency (or lag) that occurs when there's other traffic on your network. This means that your network isn't always responsive - it's wasting your time.

How does bufferbloat apply to me?

Watch the Home Internet Connections Are Unfair! (Bufferbloat) video which gives an intuitive description of Bufferbloat. Or read the more detailed Best Bufferbloat Analogy - Ever blog post.

OK - How do I get rid of Bufferbloat?

1. Measure the Bufferbloat: Use any of the tests below that measure latency both when the line is idle *and* during upload or download traffic.

- Waveform Bufferbloat Test
- Speedtest.net Test
- Cloudflare Speed Test

If the latency increases when there's traffic, you have bufferbloat. If the increase is small (less than 20-30 msec), bufferbloat is well under control. For more details about testing, read the Tests for Bufferbloat page.

2. Possible Solutions: There are lots of ways to throw time or money at this problem. Most won't work.

- Your ISP would love to sell you a faster connection, but link speed isn't the problem - it's your router buffering more data than necessary. This adds *delay* that can never be cured by faster transmission rates.
- Buying an expensive router (even one for "gaming") won't necessarily help, since many commercial, off-the-shelf router manufacturers are clueless about excess buffering in their routers.
- Twiddling the router's QoS might make a difference, but it's a hassle, and only helps a bit.

What's wrong with simply configuring QoS?

Quality of Service (QoS) settings will help, but won't solve bufferbloat completely. Why not? Any prioritization scheme works by pushing certain packets to the head of the queue, so they're transmitted first. Packets farther back in the queue still must be sent eventually. New traffic that hasn't been prioritized gets added to the end of the queue, and waits behind those previously queued packets. QoS settings don't have any way to inform the big senders that they're sending too fast/too much, so packets from those flows simply accumulate, increasing delay for all.

Furthermore, you can spend **a lot** of time updating priorities, setting up new filters, and checking to see whether VoIP, gaming, ssh, netflix, torrent, etc. are "balanced". (There is a whole cottage industry in updating WonderShaper rule sets. They all have terrible flaws, and they don't help a lot.) Worst of all, these rules create a maintenance hassle. Each new rule has to be adjusted in the face of new kinds of traffic. And if the router changes, or speed changes, or there's new traffic in the mix, then they need to be adjusted again.

Ref.: https://www.bufferbloat.net/projects/

Potential solution:
Active Queue Management
(a.k.a. queue disciplines)

AQM algorithms and Explicit Congestion Notification

Goals of AQM algorithms:

- ❑ Proactive queue control to avoid congestion
- ❑ Minimize queue delay
- ❑ Allow full utilization of the bandwidth

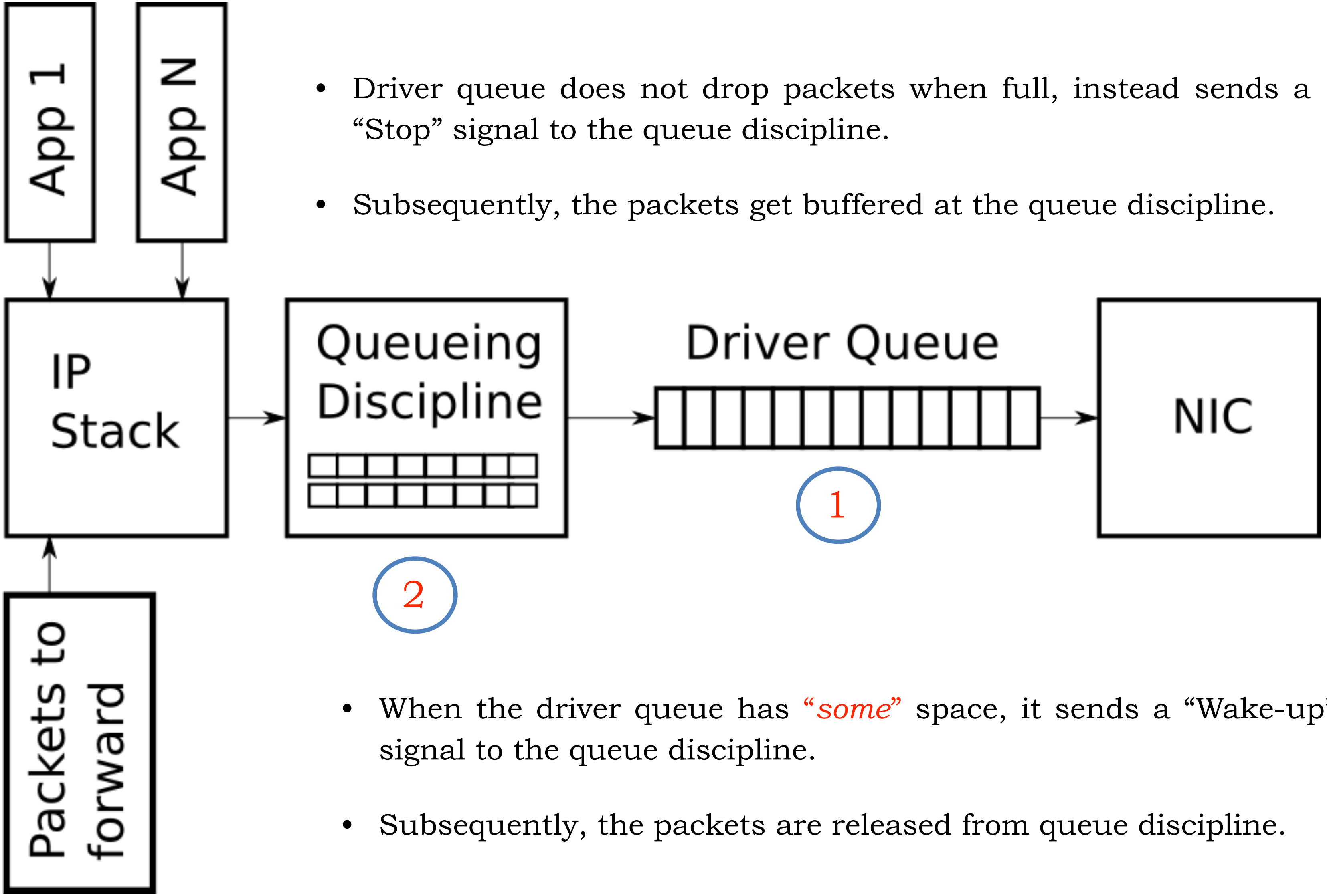
Challenges with AQM algorithms:

- ❑ Unresponsive traffic (i.e., congestion agnostic flows)
- ❑ Bursty, short lived and low volume traffic (e.g., search queries)
- ❑ Differentiating temporary burst vs persistent congestion

Explicit Congestion Notification

- ❑ Congestion signaling mechanism; marks the packet instead of dropping
- ❑ Deployed in all operating systems

Where do AQM algorithms operate?



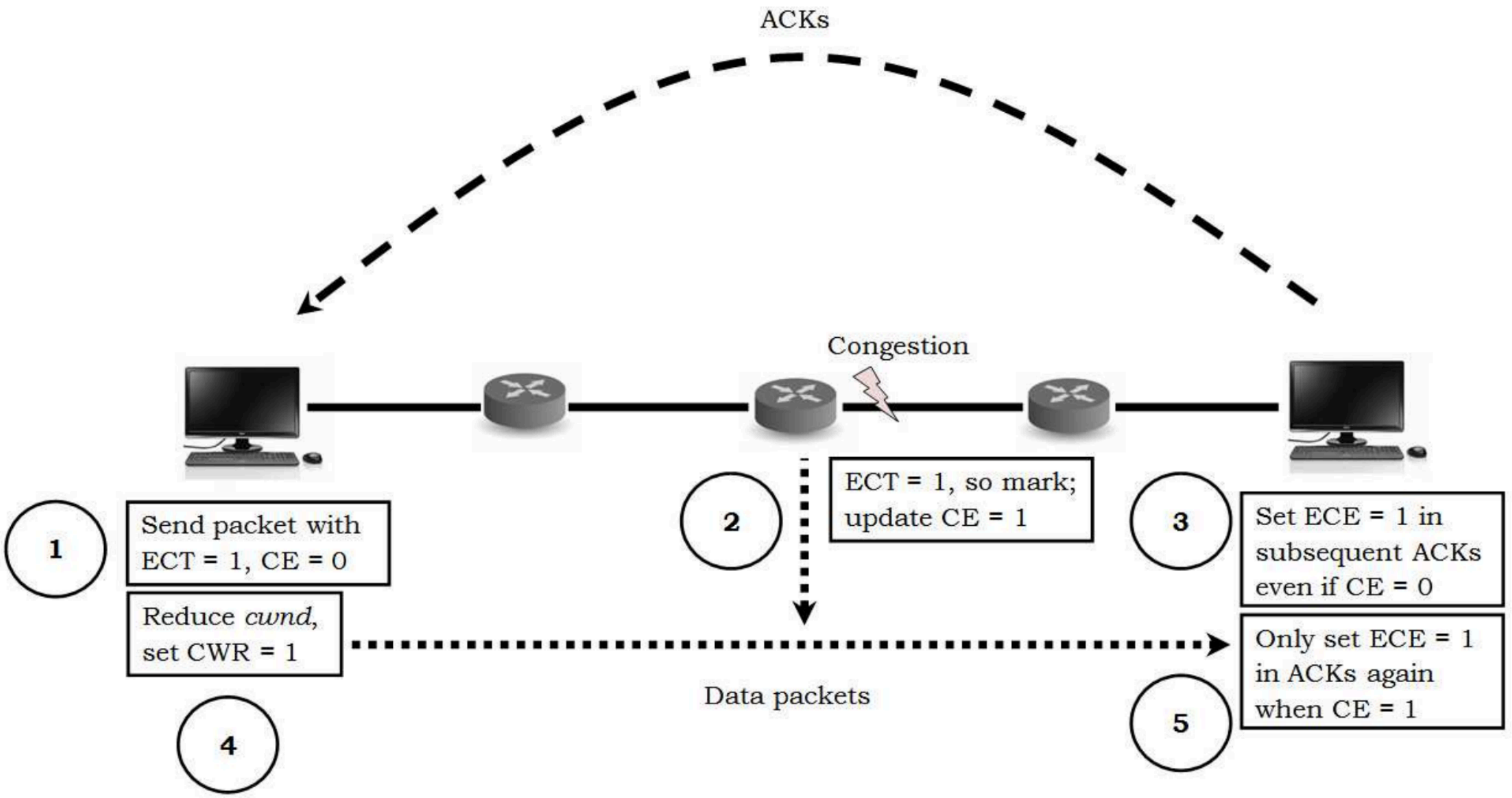
- Driver queue does not drop packets when full, instead sends a "Stop" signal to the queue discipline.
- Subsequently, the packets get buffered at the queue discipline.

- When the driver queue has "*some*" space, it sends a "Wake-up" signal to the queue discipline.
- Subsequently, the packets are released from queue discipline.

Image Source: https://www.coverfire.com/articles/queueing-in-the-linux-network-stack/

What is ECN?

Congestion signaling mechanism; uses 2 bits in IP and 2 bits in TCP headers



Controlled Delay (CoDel) [RFC 8289]

Overview:

- *Queue delay* is used as a *metric* for queue management
- Two phases: *dropping* and *non-dropping* (it starts with *non-dropping*)
- Two configurable parameters: *target* (5ms) and *interval* (100ms)

Functionality:

- Calculate current queue delay (*cur_qdelay*) for every outgoing packet

$$cur\_qdelay = dequeue\_time - enqueue\_time \qquad (1)$$

- When *cur_qdelay* > *target* for the first time, start the *interval* timer
- Enter dropping phase only if *cur_qdelay* > *target* for the entire *interval*
- Otherwise, stay in non-dropping phase because it was a short burst.
    - Reset *interval* timer and wait for *cur_qdelay* > *target* again.

Controlled Delay (CoDel) [RFC 8289]

On entering the dropping phase:

- ❑ *Drop* the outgoing packet
- ❑ Increment *count* (indicates number of packets dropped)
- ❑ Calculate time to drop the next packet using the Control Law:

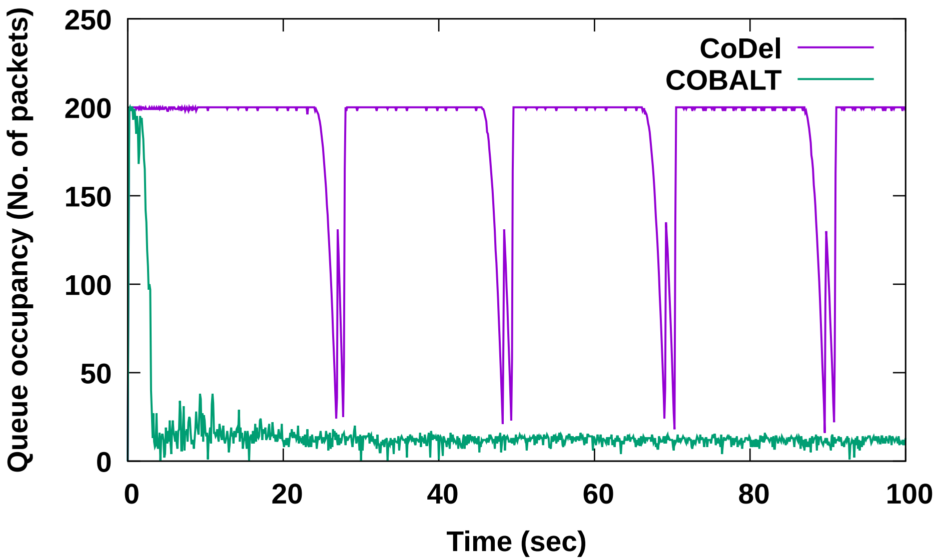
$$next\_drop\_time = current\_time + interval / sqrt(count) \quad (2)$$

While in dropping phase:

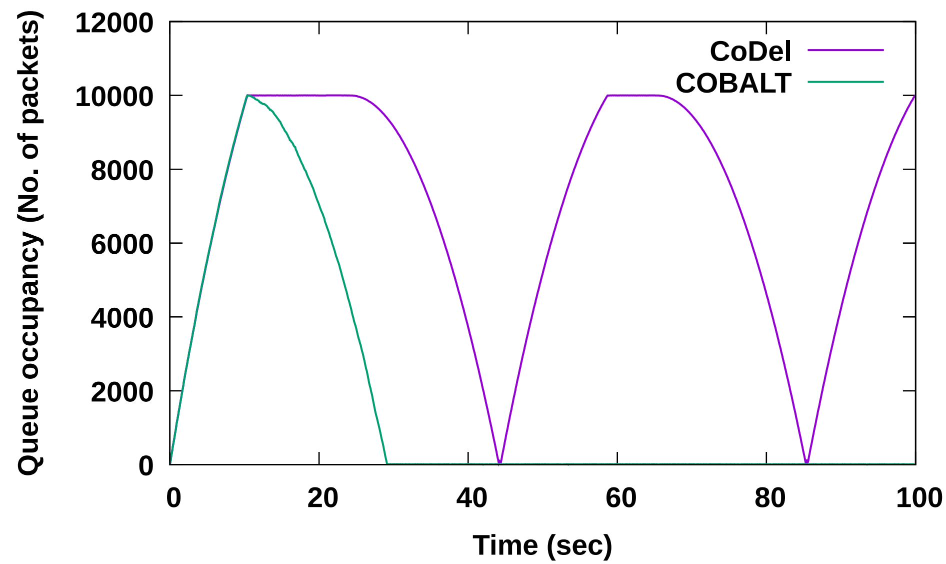
- ❑ Continue to *Drop* packets using Eq. (2) till *cur_qdelay* > *target*
- ❑ When *cur_qdelay* < *target*, exit the dropping phase
- ❑ Reset the value of *count* to 0
- ❑ Enter non-dropping phase

Known problems with CoDel

- *Queue control* is lost when some of the flows are unresponsive
  - FQ-CoDel isolates flows by providing a separate queue to every flow
  - Flow is identified by: Src IP, Src port, Dst IP, Dst Port, Protocol Number
- Discards the heuristics obtained from the dropping phase



5 TCP + 1 UDP: 200 packets queue capacity



5 TCP + 1 UDP: 10000 packets queue capacity

Palmei J, Gupta S, Imputato P, Morton J, Tahiliani MP, Avallone S, Täht D. Design and Evaluation of COBALT Queue Discipline. In 2019 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN) 2019 July (pp. 1-6). IEEE.

Proportional Integral controller Enhanced (PIE) [RFC 8033]

Overview:

- *Queue delay* is used as a *metric* for queue management
- Packets are enqueued/dropped depending on the *drop probability*
- Main configurable parameter: *target* (15ms) [a.k.a *reference queue delay*]

Functionality:

- Calculate *drop probability* every 15ms (*t_update*) as shown in Eq. (3)

$$drop\_prob = a\,(cur\_qdelay - target) + b\,(cur\_qdelay - old\_qdelay) \qquad (3)$$

  - *a* and *b* are PI control parameters
  - *cur_qdelay* is an estimate of current queue delay

- The incoming packet is dropped if *drop_prob* > *R*, otherwise enqueued
- R is a random number between 0 and 1.

Proportional Integral controller Enhanced (PIE) [RFC 8033]

Two ways to estimate current queue delay (*cur_qdelay*)

- ❑ *Little's Law* as shown in Eq. (4) [recommended default in RFC 8033]

$$cur\_qdelay = cur\_qlength \, / \, avg\_departure\_rate \qquad (4)$$

- ❑ Using timestamps like CoDel, as shown in Eq. (1)

$$cur\_qdelay = dequeue\_time - enqueue\_time \qquad (1)$$

Other notable features in RFC 8033:

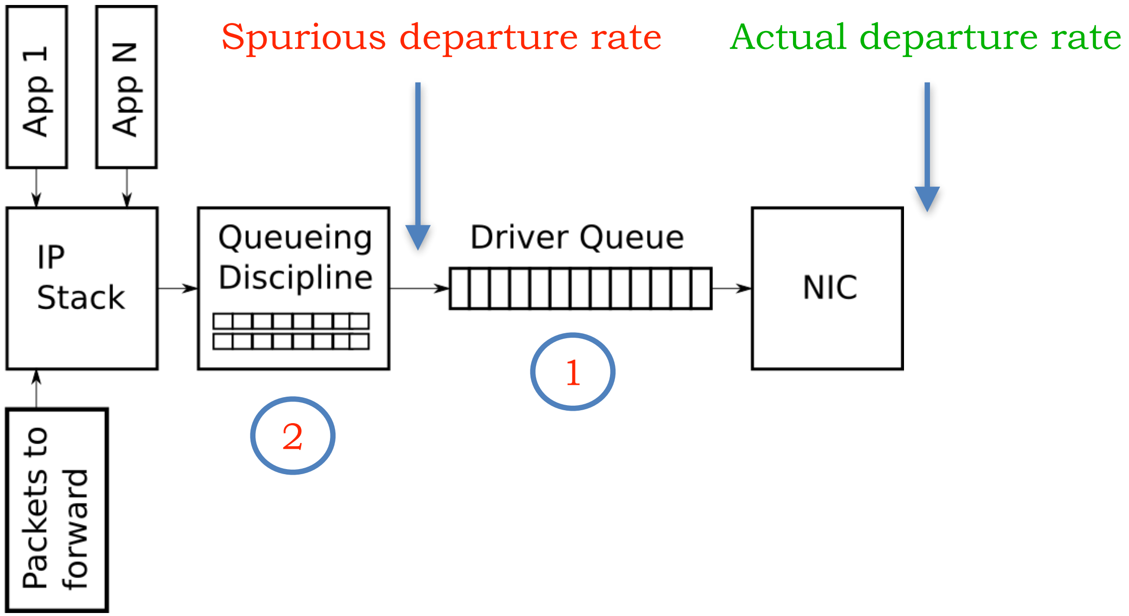
- ❑ Burst allowance (allows small bursts to pass by without getting punished)
- ❑ Auto-tuning the *drop_prob*
- ❑ Avoids a sharp rise in *drop_prob*
- ❑ Decays *drop_prob* when queue is idle
- ❑ Activating / deactivating PIE depending on current queue length

PIE: Implementation issue in the Linux kernel

Problem:

❑ *Spurious departure rate* affects the calculation of *cur_qdelay* (See Eq. (4))

$$cur\_qdelay = cur\_qlength \,/\, avg\_departure\_rate \tag{4}$$

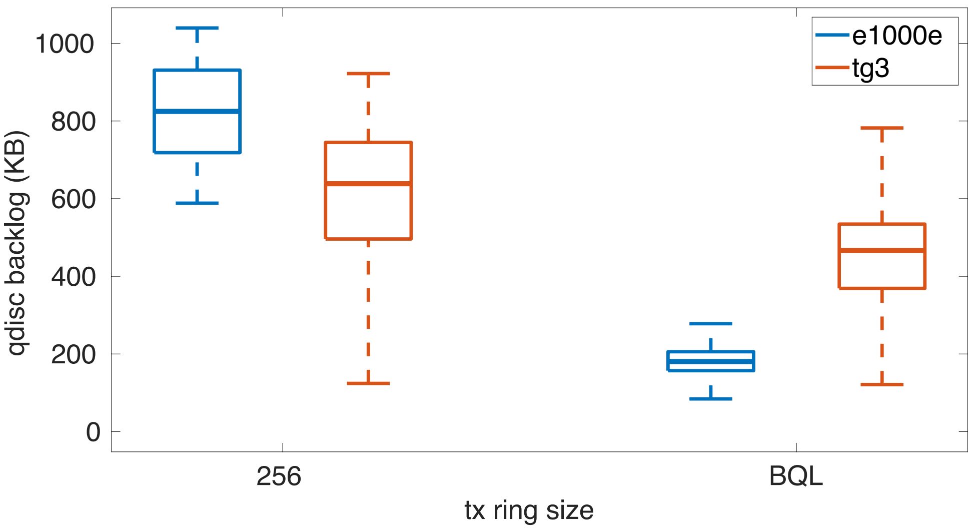


Solution:

❑ Use *timestamps* to calculate *cur_qdelay* in PIE, like CoDel does:

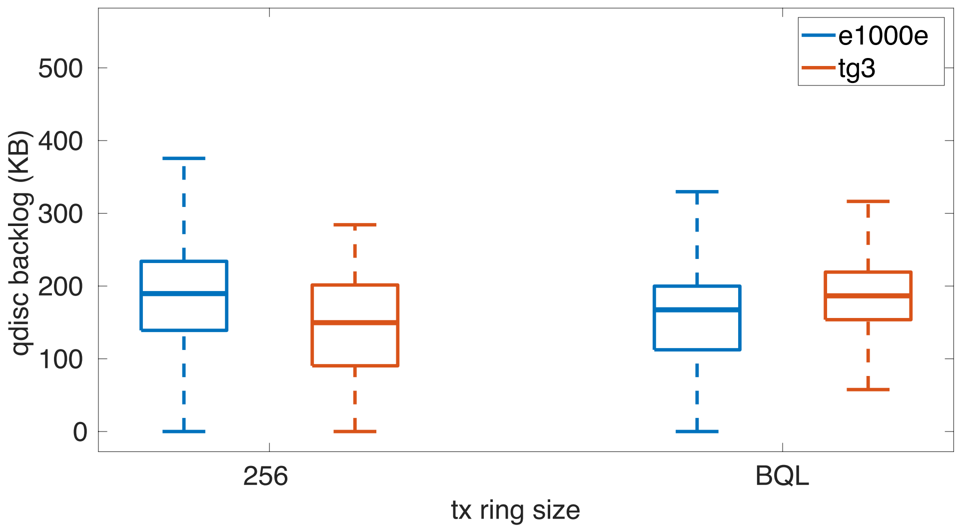
$$cur\_qdelay = dequeue\_time - enqueue\_time \tag{1}$$

PIE: Implementation issue in the Linux kernel (fixed)



When Little's Law is used in the Linux kernel.

Expected disc backlog: 180 KB



When timestamps are used in the Linux kernel.

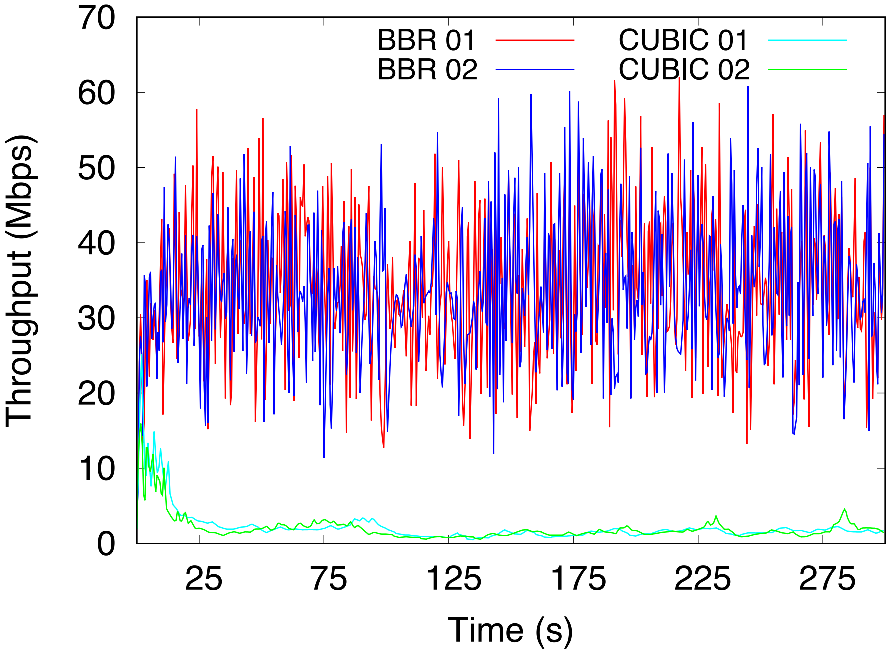
Expected disc backlog: 180 KB

Imputato P, Avallone S, Tahiliani MP, Ramakrishnan G. Revisiting design choices in queue disciplines: The PIE case. Computer Networks. 2020 Apr 22;171:107136.

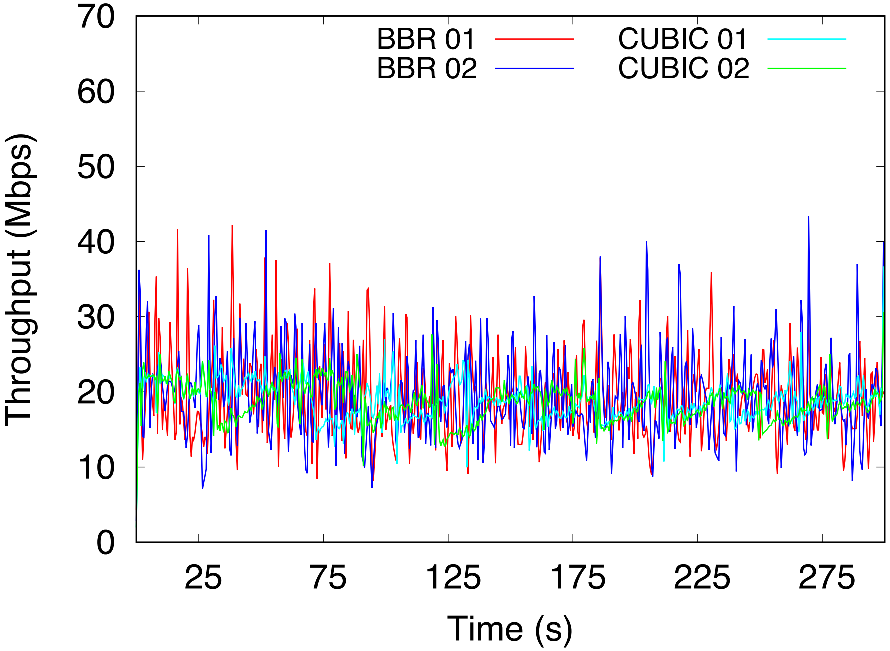
Flow Queue PIE (FQ-PIE)

Isolates flows by providing a separate queue to every flow

- ❑ Same as FQ-CoDel, except that individual queue runs PIE instead of CoDel
- ❑ Introduced in Linux 5.6 (March 2020)



PIE



FQ-PIE

Ramakrishnan G, Bhasi M, Saicharan V, Monis L, Patil SD, Tahiliani MP. FQ-PIE Queue Discipline in the Linux Kernel: Design, Implementation and Challenges. In2019 IEEE 44th LCN Symposium on Emerging Topics in Networking (LCN Symposium) 2019 Oct 14 (pp. 117-124). IEEE.

New Internet Draft on FQ-PIE (version-00)

Your feedback is much appreciated!

Workgroup: Transport and Services Working Group
Internet-Draft: draft-tahiliani-tsvwg-fq-pie-00
Published: 8 November 2024
Intended Status: Experimental
Expires: 12 May 2025
Authors: M. P. Tahiliani
National Institute of Technology Karnataka

Flow Queue PIE: A Hybrid Packet Scheduler and Active Queue Management Algorithm

Abstract

This document presents Flow Queue Proportional Integral controller Enhanced (FQ-PIE), a hybrid packet scheduler and Active Queue Management (AQM) algorithm to isolate flows and tackle the problem of bufferbloat. FQ-PIE uses hashing to classify incoming packets into different queues and provide flow isolation. Packets are dequeued by using a variant of the round robin scheduler. Each such flow is managed by the PIE algorithm to maintain high link utilization while controlling the queue delay to a target value.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://mohittahiliani.github.io/draft-tahiliani-tsvwg-fq-pie/draft-tahiliani-tsvwg-fq-pie.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-tahiliani-tsvwg-fq-pie/.

Discussion of this document takes place on the Transport and Services Working Group Working Group mailing list (mailto:tsvwg@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/tsvwg/. Subscribe at https://www.ietf.org/mailman/listinfo/tsvwg/.

Source for this draft and an issue tracker can be found at https://github.com/mohittahiliani/draft-tahiliani-tsvwg-fq-pie.

Low Latency Low Loss Scalable Throughput (L4S)

L4S has three main components

Scalable Congestion Control (Prague?), AQM (Step marking) & Modified ECN

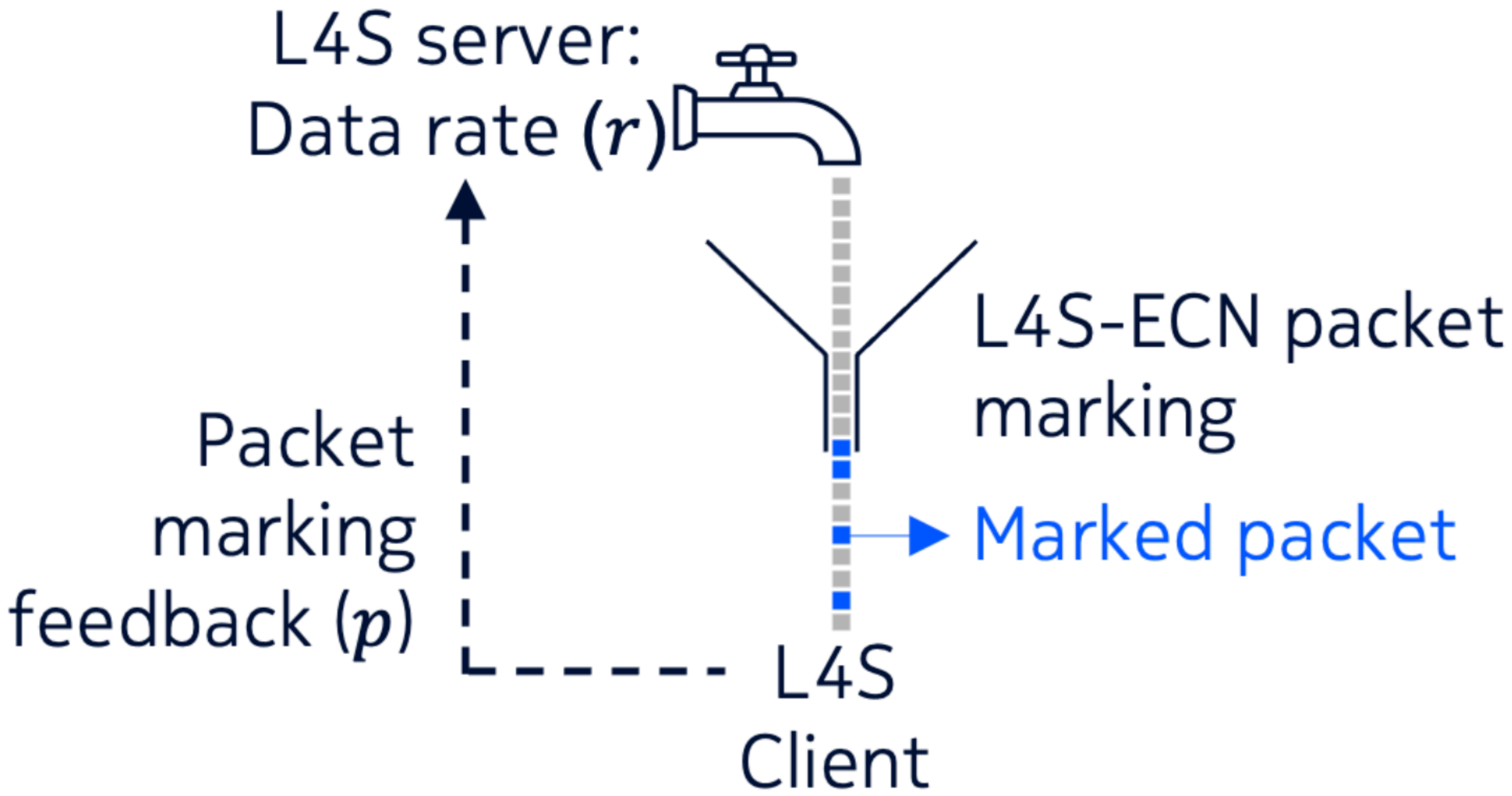


Image Source: https://www.bell-labs.com/research-innovation/projects-and-initiatives/l4s/

What does L4S offer? Where does it fit?

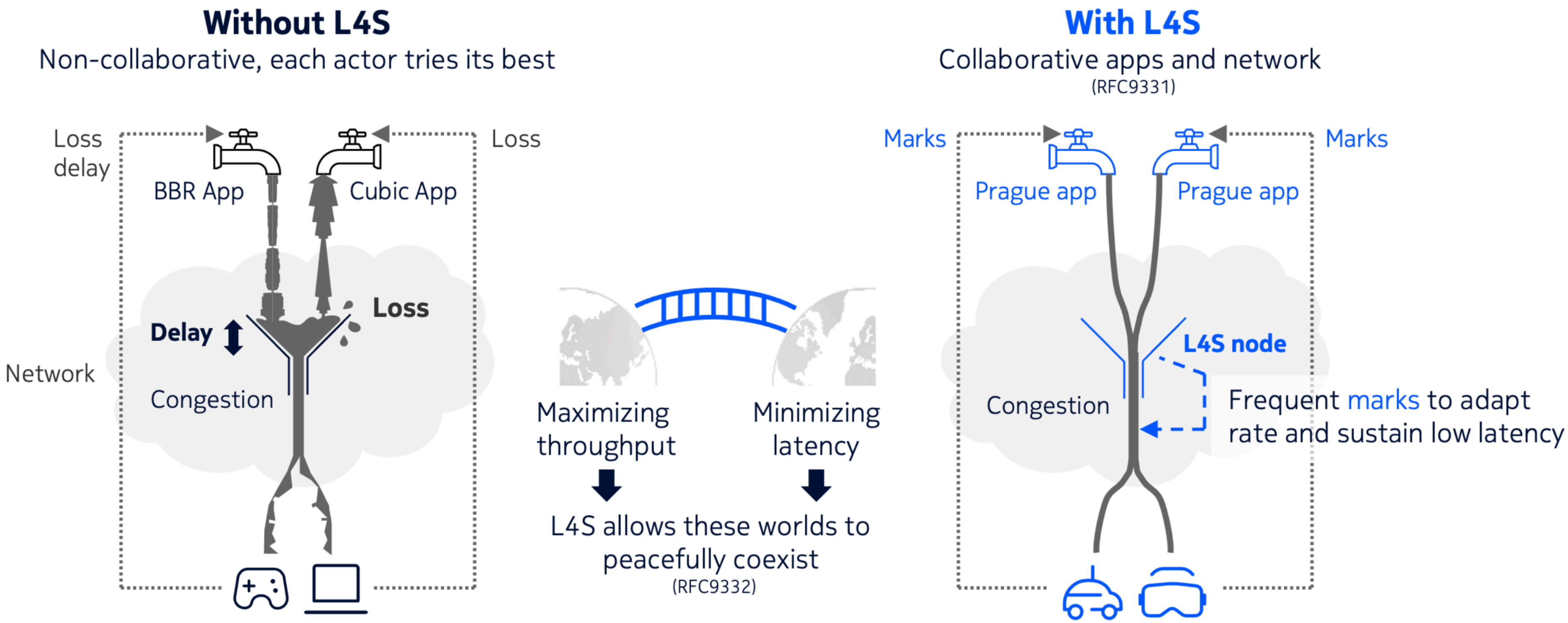


Image Source: https://www.bell-labs.com/research-innovation/projects-and-initiatives/l4s/

Current State of Deployment of Bufferbloat Solutions

AQM algorithms:

- FQ-CoDel: most widely deployed and default in Linux distros
- FQ-PIE: available in Linux and used as default by some Linux distros
- Random Early Detection (RED) and its variants (ARED, WRED, etc)

Smart Queue Management:

- CAKE: Common Applications Kept Enhanced - part of Linux mainline
    - AQM, ACK filtering, Bandwidth Shaper, DRR+ packet scheduler

LibreQoS (https://libreqos.io/):

- Leverages FQ-AQM to improve QoS for ISPs and end users

L4S (Prague, BBRv3):

- Under active deployment by end user facing orgs (Apple, Samsung, etc)

Thank you!