Discussion Paper

Per-Flow Scheduling and the End-to-End Argument

Bob Briscoe*

22 Jul 2019

Abstract

The primary message of this paper is that rough equality between flow rates is only needed in times of famine (when congestion is high). It is a common misunderstanding to interpret this guidance for end-to-end transports as a requirement that the network ought to enforce precisely, whether in times of plenty or famine. In times of plenty, unequal flow rates are a feature not a bug. This puts the end system in control, allowing innovation without asking the network's permission.

This paper was written now because two approaches have been proposed to enable end-systems to maintain extremely low queuing delay: L4S and SCE. Without explaining the abbreviations here, the high level point is that they both compete for the same last ECN codepoint in the IP header. SCE seems to require per-flow scheduling in the network in order to provide benefit. Whereas L4S provides benefit either with per-flow scheduling or with a DualQ Coupled AQM.

One of the reasons that the DualQ approach was developed was so that those who want extremely low latency would not be forced to have to use per-flow scheduling (but they still could if they wanted to). To those who see per-flow scheduling as a panacea, it does not seem important to allow this choice. That is why it was thought necessary to explain the concerns that people have about per-flow scheduling.

CCS Concepts

•Networks → Network Architecture, Network algorithms;

Keywords

Data Communication, Networks, Internet, Control, Congestion Control, Scheduling, Quality of Service, Performance, Latency, Algorithms, Network Architecture,End-to-End Argument, Active Queue Management, AQM, Explicit Congestion Notificiation, ECN

*research@bobbriscoe.net,

1 Introduction

This paper is about scheduling the packets of individual application flows (termed just 'flows' in this paper). It concerns the choice between scheduling between flows in the network or in a distributed way by the end-to-end transport layer. Even though the end-to-end approach is a form of scheduling, we will solely use the term 'per-flow scheduling' to imply 'in the network', which is its normal usage.

Coupled DualQ AQMs [DSBEAT19] were developed so that extremely low (sub-millisecond) queuing delay could be achieved without per-flow scheduling. DualQ AQMs use the L4S Low Latency Low Loss Scalable throughput (L4S) definition [DSBET19] of the last remaining ECN codepoint in the IP header, which is also compatible with per-flow scheduling.

An alternative approach with similar goals has been proposed called Some Congestion Experienced (SCE [MT19]). It competes for the same ECN codepoint.

SCE-ECN requires per-flow scheduling to provide benefit.[1] In contrast, L4S-ECN supports either per-flow scheduling or the DualQ approach.

One of the reasons that the DualQ approach was developed was so that those who want extremely low latency would not be forced to use per-flow scheduling. To those who see per-flow scheduling as a panacea, it does not seem important to allow a choice. Therefore, this paper explains the concerns that people have about per-flow scheduling.

This paper could also be seen as a set of arguments against deploying per-flow scheduling at all. That boat has already sailed, given the widespread deployment of FQ_CoDel since about 2017. The intent is solely that "the market (not the IETF) can

[1] An attempt to show that SCE can provide benefit without per-flow queues [MH19] it still meant to do per-flow scheduling. (Not relevant to this paper but, at the time of writing, it is questionable whether it schedules as intended anyway.)

choose" between per-flow scheduling by the network and by end-systems.

Some of the concerns with per-flow scheduling are because it runs counter to the commonly accepted way that networks are designed—what might be called the accepted architectural norms in networking—particularly layering. That is not such a concern if per-flow-scheduling is one of many ways to provide extremely low latency. It would be of great concern if SCE were to use the last ECN codepoint so that per-flow scheduling became the only way.

This is the purpose of the end-to-end argument [SRC84] — not to say "Thou shalt not," about embedding functions in the network; rather to say "Are you sure there is not a an end-to-end way of doing that, which will allow the network to remain more generic for innovative new behaviours?"

2 Rate Inequality is Desirable

Flow rate inequality is a feature not a bug. Equality of flow rates has been a rough goal of transport protocols. But it is a misconception to translate that into a requirement for the network to enforce precise equality. The 'rough' aspect of the goal is more important than the equality. As we shall see, giving applications and transports no wriggle room means they cannot innovate without the permission of the network, which is the teaching of the end-to-end argument.

To be clear, we are talking here about the relative rates of all the flows of one user, or one household, or perhaps one small business. Inter-customer rate limits are not in question here; only intra-customer.

In 2007–8, a debate raged in the research community and the IETF, triggered by the provocative paper entitled "Flow Rate Fairness: Dismantling a Religion" [Bri07]. That paper made the point that rate equality had no basis as a goal. The unfortunate choice of the word 'fairness' has led generations coming new into networking to think that rate equality really is fair in some sense. Whereas generally accepted analysis (e.g. Kelly's model linking congestion control to economics [KMT98], or Doyle's similar work) has shown that a fair allocation in the economic sense would likely involve significantly unequal rates. [Bri07] made the point that fairness can be judged on other criteria than economics, but that economics is the best discipline to judge fairness for a global scale system, as long as other views of fairness can be applied within that. [Bri07] threw down a challenge for someone to justify the goal of flow rate equality. Floyd and Allman responded, publishing on the IETF's independent stream [FA08]. They argued that, although unequal rates might be optimal, rate equality was simpler[2], particularly when ensuring that no one flow starves any more than any other

The important take-away from that episode in history is that flow rate equality is not a goal to be enforced in the network, except perhaps at high levels of congestion. It would be presumptuous to say that has become the accepted view. It would be more accurate to say that those involved in congestion control design are converging on consensus around this view.

A corollary is that rate equality should only be a 'famine response'—should only really be important if congestion is high. This is recognized in end-to-end approaches like BBRv2, in which it has been proposed to only respond to loss above a threshold level[3].

As capacity is becoming more plentiful, it would be a great shame if we introduced widespread enforcement of equality between the rates of one customer's flows.[4]

For instance, without per-flow equality enforcement, someone with 400 Mb/s to and from their home could enjoy a VR/AR application that used say 60–80% of their capacity and still have 80–160 Mb/s available for everything else. As long as the VR app adapted if absolute levels of congestion became high (e.g. many TCP flows competing, or two VR sessions), there would be no starvation. This example is not given as exceptional. One assumes it will be normal in future for many activities, e.g. online shopping, to be possible over a VR platform.

In contrast, with per-flow scheduling enforcement, every time one or two flows in the same home as the VR app transferred a reasonably large amount of data, even briefly, per-flow scheduling would cut down the VR/AR app to 200 Mb/s or 133 Mb/s for the duration of the other flow(s).

Then, to make the VR app deployable, some way to request an exception to per-flow scheduling would have to be deployed. A WiFi policy enforcement approach called PoliFi is proposed in Høiland-Jørgensen's thesis [HJ18]. However, that would

[2] There was a misunderstanding about the simplicity of mechanisms needed for flow-rate inequality—just weighted congestion controls. But there's no need to labour that point here.

[3] A more gradual threshold would be preferable, but again, that's straying into secondary arguments

[4] We can be relaxed about the large deployment of FQ_CoDel, which is on software routers that can be revisited if needed.

not be sufficient if the user's ISP (or the ISP of a remote peer) had deployed per-flow scheduling into the downlink of the user's access—an approach akin to RSVP signalling would be needed. Alternatively, per-flow scheduling would have to be subverted using multiple flow IDs and stitching packets back into order later. If the flow rate equalizing aspect of per-flow scheduling was routinely overridden with multiple flows, it would have become an irritating encumbrance, just making application programming unnecessarily complicated.

Once the network intervenes to do the job that end-systems are capable of (allocating themselves bandwidth and pacing) the complexity consequences snow-ball. The point of the end-to-end argument is not to follow it blindly but, if you need to violate it, it reminds you to think carefully about the consequences—you're probably doing something you (or those who come after you) will regret.

2.1 Unequal average flow rates

2.1.1 Elastic Flows

As a thought experiment, consider the example elastic flows in Figure 1a) with a pure 'fair' queuing scheduler, where the dashed horizontal line represents the maximum capacity. The figure shows a few relatively small flows using the capacity in an idealized schematic (not showing the dynamics—just instantly getting up to speed or cutting speed). There is only one period of contention between flows, due to the slightly longer third (light green) flow, which shares the capacity with the fourth (dark blue) flow.

Figure 1b) shows how the same flows would be treated by the scheduler if a long-running flow were added to the mix. The scheduler halves the rate and therefore doubles the completion time of the first two short flows. Once the scheduler forces the light green flow to halve its rate, there is a knock-on effect where the light green flow contends with all the remaining short flows and they, in turn, contend with the next. So all the later flows take 3 or 4 times longer to complete (in this example).

Figure 1c) shows the outcome of a different arrangement of the same flows. Here the larger each flow's volume the lower its weight. For the point we're trying to make, it doesn't matter whether a weighted priority scheduler or weighted congestion controllers are being used. The point is that these highly unequal flow rates have led to flow completion times considerably better than with equal flow rates, indeed nearly as good as they were before the contending long-running flow was introduced.

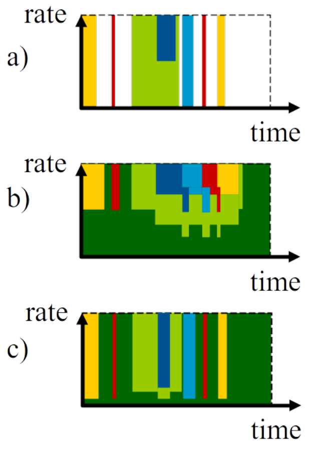

Figure 1: Is Equal Flow Rates a Valid Goal?

Even the longest-running (dark green) flow should complete at about the same time. This is because the area of each plot represents the amount of bytes in each flow (rate×time), so the sum of their areas is the same whether they are slower and last longer, or faster and complete sooner.

The inefficiency of the dynamics cannot be ignored, but the purpose of this thought experiment is to undermine the idea that equality between flow rates is a desirable goal, or indeed 'fair'.

This is not a new idea; it is an translation of Shortest Remaining Processing Time scheduling into bandwidth scheduling. [GM02] showed that, with a heavy-tailed flow-size distribution, response times can be improved by an order of magnitude. Even without foresight of the length of all the flows, it is possible to start high then decay the weight the more bytes there are in a flow. PIAS [B+15] is a practical realization of the approach for data centres that steps down a set of priority queues based on the number of bytes sent per-flow. It can roughly halve flow completion time for short flows relative to DCTCP or L2DCT.

However, on the public Internet, such a scheduler in the network would harm less-elastic applications (e.g. streaming or conversational video), which would require exceptions to be allowed, which would open a can of management and security worms. In other words, such a scheduler is not generic, and therefore not advisable to be embedded in the network, by the end-to-end argument.

In contrast, a similar e2e solution would be more feasible. To my knowledge, three variants of TCP have been proposed that decay from higher to lower aggressiveness as a flow proceeds [Sd02, ZTH04, MSSM12]. Only the last of these ends up less aggressive than a TCP-Friendly flow, but unfortunately the authors have not investigate this aspect of the implementation. There is an incentive for

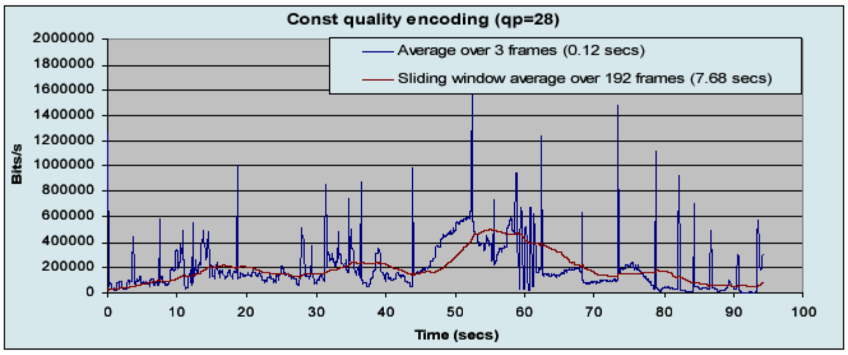

Figure 2: Bit Rate Variation of an Example Constant Quality Video

long flows to be less aggressive — to improve the completion time of a user's own shorter flows. Because this is an e2e solution, in any scenario where relaxing aggression is not appropriate, the end-system doesn't have to, e.g. not for video, and not if throughput continually reduces (which would imply competition with an incompatible long-running flow with constant aggressiveness).

Also, the network-based solution would only work where it was deployed, whereas the e2e solution would work for its user anywhere (except where per-flow scheduling is deployed.

2.1.2 Scavenger Flows

Note that the lower weight higher volume thought-experiment introduced in § 2.1.1 was applicable to flows of the *same* importance. Less than best efforts (LBE) or 'scavenger' service is intended traffic with a less important deadline.[5]

LBE can be implemented either with priority scheduling at a bottleneck link or end-to-end using an LBE congestion control [RW13]. Nonetheless, as above, network-based LBE would only work where it was deployed, whereas e2e LBE works without any special network equipment...

...Except certain special network equipment, namely per-flow scheduling, prevents e2e protocols from scavenging capacity.[6]

2.2 Unequal dynamic flow rates

The information rate arriving at the human eye, or at a video camera, varies continually and widely depending on the complexity of the scene (Figure 2).

[5] Which could still use an even lower weight for larger flows.

[6] Any AQM that cuts queuing delay significantly confounds delay-based scavenger protocols [ASAB17], but per-flow scheduling unnecessarily confounds non-delay-based scavenger protocols as well, such as 4CP [LVG07], MulTFRC [DW11] or DA-LBE [HRPA17].

Despite this, video is often packaged into constant bit-rate containers—whether by DVD, or communication link—but that's for the distributor's convenience. Then the quality of complex scenes (the most complex being rapid uncoordinated motion such as moving leaves or water, sports action, remote viewing via a drone) has to be degraded, clipping the bit-rate peaks to fit into a set size of pipe.

Studies have repeatedly found that, if quality varies, humans perceive the quality of the whole video as if it was close to the minimum quality, not the average. Consequently, increases in network capacity have been exploited to use scalable video coding, e.g. as available in x264, to maintain near-constant quality but varying bit-rate. When live interactive video is being transferred the variations cannot be buffered without harming latency. It is easier to hide the degradation in quality from human perception when short spikes of bit-rate are clipped. However, flattening the longer term bit-rate variations is much more noticeable as quality degradation.

When a variable bit rate (VBR) video shares a FIFO with an elastic flow, the elastic flow moulds itself around the peaks and troughs of the video, and the video responds to the resulting congestion on a slower time-scale to ensure that their rates are roughly equal.

In contrast, with a per-flow scheduler, whenever the video consumes more than half the capacity (or 1/n with n-1 other flows), the scheduler only allows the video 1/n of the capacity. During periods when it needs more than 1/n, the focused squeezing of the video flow by the scheduler rapidly grows its per-flow queue, and any AQM applied to that queue introduces sufficient loss (or ECN) to cause the video codec to adapt down to a lower quality level. In a brief trough below 1/n of capacity, the codec will stick at lower quality. In a longer trough it will adapt up. So the per-flow scheduler either causes the codec to unnecessarily use a lower quality codec, or it shaves off the quality of the more complex, higher bit rate scenes [TE+15, Figs 3.15 & 3.16]. Thus, instead of constant quality, the video experiences high levels of loss, even though the other flows would temporarily yield if left in control.

The effect is similar when the throughput of two or more interactive VBR videos is high enough to compete for capacity. With a FIFO queue they multiplex more efficiently together when peaks meet troughs, and the end systems adapt to the congestion when peaks meet peaks [MANC09]. However, with a per-flow scheduler, both (all) videos frequently lose quality during their more complex higher bit rate scenes.

2.3 Policy vs. Mechanism

The problem with using per-flow scheduling to isolate latency can be characterized as embedding policy where there should only be mechanism. Separating each flow into its own queue or bucket achieves inter-flow latency isolation, but then flows have to be merged back together into the link. So something has to decide what share of the capacity each flow gets. The introduction of bandwidth allocation between flows raises controversial policy questions—no choice is neutral.

Rough consensus around an equal rate policy could probably be achieved for a capacity famine scenario. But during normal times of plenty, users and applications will want other policies—as we have seen above.

So far, with FIFO queues, the intra-customer tussle between applications for a customer's capacity has not been a great problem for the Internet. As the saying goes, if it ain't broke, don't fix it. And certainly be wary of involving the network in mediating that question—that *would* raise genuine net neutrality concerns.

Although it ain't broke, it's worth having a solution in reserve in case it breaks. For the task of latency isolation, a policer is a more appropriate policy tool than a scheduler. The per-flow queue protection function (QProt [BW19]) introduced into DOCSIS to be able to protect the Low Latency queue is a good example. It protects a shared queue from any flow that is about to cause the delay to exceed a target.

The DOCSIS QProt function does not interfere with capacity allocation, except inasmuch as it attributes blame for excess queuing in proportion to the share of capacity being used when the queuing occurs. As long as the queue remains shallow, flows have a large degree of wriggle-room to use different shares of the customer's capacity.

The feature that distinguishes per-flow QProt from per-flow scheduling is that, under regular conditions, low latency still works without QProt (much as TCP still shares out capacity without per-flow rate policing). Indeed, if it turns out not to be needed, DOCSIS operators can disable QProt, and DOCSIS will still give low latency. Also the L4S reference implementation in Linux does not use queue protection, instead using a simpler form of bulk overload protection.

The next section criticizes per-flow scheduling for violating layering. Admittedly that's hypocrisy, because per flow queue protection similarly violates layering. We can only admit that we're guilty as charged. And say in our defence that one crime is better than two. In mitigation of the remaining crime, if flow IDs are inaccessible at a lower layer (e.g. in vRANs), L4S can still provide low latency queuing but SCE cannot work at all. The consequence of not being able to deploy queue protection (e.g. in vRANs) would be that in anomalous conditions (whether by accident or malice) queue delay could become greater than was intended.

3 Layering

3.1 Encapsulation

Per-flow scheduling requires access to layer-4 flow identifiers to be effective. It ought to be possible to implement AQM at any layer. However, per-flow scheduling does not satisfy this requirement, because network encapsulation is not designed to ensure layer-4 headers are accessible.

Many access networks are effectively layer-2 subnets. For instance, in the 5G radio access network (RAN), IP headers are compressed, ciphered and encapsulated by Packet Data Convergence Protocol (PDCP) headers then Radio Liink Control (RLC) headers. The architecture of the 5G RAN includes a facililty for a virtual RAN (vRAN). In a vRAN PDCP encapsulation occurs remote from the gNodeB (the 5G base station). However, AQM would need to be applied at the gNodeB, which will often be the bottleneck. So in the vRAN case, visibility of layer-4 headers would be impossible. 5G equipment is not intended to scale to the alternative of giving each layer-4 flow a separate 5G QoS flow identifier (QFI) (typically QFIs are used for aggregates of flows, e.g. all Internet traffic is usually within a single QFI).

In contrast, an AQM is intended to be able to apply ECN-marking at layer-2. [Joh17] is a proposal to add an ECN-capability at the RLC layer, which is part of the programme of work to add the ECN capability to those lower layer protocols that need it [BKT19], as has already been done with MPLS, TRILL and various tunnelling protocols.

If the IETF made per-flow scheduling mandatory in order to support extremely low latency with high throughput, it would effectively send a message to the 3GPP that the IETF did not care if its low latency technology did not work for 3GPP networks, especially given the 3GPP RAN has correctly followed network layering principles.

3.2 Privacy

If a VPN at layer-3 or lower is used, e.g. using IPSec, the layer-4 flow IDs are concealed from per-flow scheduling.

Therefore per-flow scheduling treats all the flows in a VPN as a single flow, giving the whole VPN 1/n of the capacity when there are n-1 other flows at the bottleneck, even if there are many flows within the VPN.

IPSec conforms to the commonly agreed layering practices of the Internet architecture. The problem here is that per-flow scheduling does not. So if someone wants both high privacy and low latency, they cannot. Certainly, they could still have privacy by using a layer-4 VPN, but such privacy is more open to traffic analysis. It is not our place to tell people who are concerned about privacy that they don't need strong privacy.

The L4S architecture was developed to give a really high performance solution that was also compatible with network layering. Then people don't have to choose between the best privacy and the best performance—they can have both.

3.3 Transport Protocol Evolution

Layering is meant to decouple evolution of higher layer protocols from the constraints of compatibility with lower layers. Strictly, per-flow scheduling fails on this point. However, it is a rather picky point, because per-flow scheduling only requires access to the first 32 bits of the transport layer, which are already widely accessed in the network by firewalls, load balancers, equal cost multipath routing, etc. Thus, these 32 bits have already become a *de facto* endpoint addressing sub-layer of the transport layer that is common across all transport protocols.[7]

[7] With the Encrypting Security Payload of IPSec, the 32-bit Security Parameters Index sits in this position, but it is common to all flows within the same security association.

4 Virtual Queues

The virtual queue (VQ) approach sacrifices a small proportion of capacity (e.g. 1%) to keep the real queue empty most of the time (i.e. just containing the packet being serialized).

A VQ represents how long the queue would be if the drain rate were slightly slower. It is a number representing the VQ's length, that is added to whenever a packet is enqueued and subtracted from slightly faster than the real queue [Ear09, AKE+12, Bri17]. An AQM can then ECN-mark packets in the real queue based on the length of the VQ, to keep the average arrival rate just below capacity.

Johansson has simulated L4S with a VQ in a 5G environment with varying capacity (see Annex of [Joh17]). Average and 95-%ile queueing delay were an order of magnitude lower than with L4S alone, which in turn was an order of magnitude lower than Cubic. A VQ is likely to be necessary to mitigate queuing due to radio capacity decreases, which is also demonstrated in Johansson's simulations.

Low Latency DOCSIS [WSB19] includes a VQ. It is currently recommended to be set at the same rate as the real queue, but in future operators will be able to dial it down to reduce queuing further if they choose to trade off a little capacity.

It is likely that the VQ approach will become more prevalent as capacity continues to become more plentiful (but time does not).

In contrast, per-flow scheduling becomes ineffective with a VQ, because the VQ usually leaves only one real packet to choose from—perhaps two or three at most. The irony of using per-flow scheduling to reduce queuing delay is that it needs a queue to work. Whereas a VQ exploits the collective abilities of all the senders to distribute scheduling and pacing, with minimal buffering and minimal function in the network.

5 Implementation Complexity

The DualQ AQM was designed with the aim of shifting as much of the function of an AQM as possible to end-systems. Specifically, the low latency side of the AQM shifts smoothing the queue and scheduling out of the network.

No matter how much the implementation of per-flow scheduling can be simplified, it seems unlikely it can be simpler than not doing it at all.

Also, as we have already pointed out, ways will be needed for applications to request exceptions to per-flow scheduling, which will also add further to whole system complexity.

References

[AKE+12] Mohammad Alizadeh, Abdul Kabbani, Tom Edsall, Balaji Prabhakar, Amin Vahdat, and Masato Yasuda. Less Is More: Trading a Little Bandwidth for Ultra-Low Latency in the Data Center. In *Proc. USENIX Symposium on Networked Systems Design and Implementation (NSDI'12)*, April 2012.

[ASAB17] Rasool Al-Saadi, Grenville Armitage, and Jason But. Characterising LEDBAT Performance Through Bottlenecks Using PIE, FQ-CoDel and FQ-PIE Active Queue Management. In *Proc. IEEE 42nd Conference on Local Computer Networks (LCN)*, pages 278–285, October 2017.

[B+15] Wei Bai et al. Information-Agnostic Flow Scheduling for Commodity Data Centers. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 455–468, Oakland, CA, 2015. USENIX Association.

[BKT19] Bob Briscoe, John Kaippallimalil, and Pat Thaler. Guidelines for Adding Congestion Notification to Protocols that Encapsulate IP. Internet Draft draft-ietf-tsvwg-ecn-encap-guidelines-13, IETF, May 2019. (Work in Progress).

[Bri07] Bob Briscoe. Flow Rate Fairness: Dismantling a Religion. *ACM SIGCOMM Computer Communication Review*, 37(2):63–74, April 2007.

[Bri17] Bob Briscoe. The Native AQM for L4S Traffic. Technical Report TR-BB-2017-002; arXiv:1904.07079 [cs.NI], bobbriscoe.net, September 2017.

[BW19] Bob Briscoe and Greg White. Queue Protection to Preserve Low Latency. Internet Draft draft-briscoe-q-protection-00, IETF, July 2019. (Work in progress).

[DSBEAT19] Koen De Schepper, Bob Briscoe (Ed.), Olga Albisser, and Ing-Jyh Tsang. DualQ Coupled AQM for Low Latency, Low Loss and Scalable Throughput (L4S). Internet Draft draft-ietf-tsvwg-aqm-dualq-coupled-10, IETF, July 2019. (Work in Progress).

[DSBET19] Koen De Schepper, Bob Briscoe (Ed.), and Ing-Jyh Tsang. Identifying Modified Explicit Congestion Notification (ECN) Semantics for Ultra-Low Queuing Delay (L4S). Internet Draft draft-ietf-tsvwg-ecn-l4s-id-07, IETF, July 2019. (Work in Progress).

[DW11] Dragana Damjanovic and Michael Welzl. An Extension of the TCP Steady-state Throughput Equation for Parallel Flows and Its Application in MulTFRC. *IEEE/ACM Transactions on Networking*, 19(6):1676–1689, December 2011.

[Ear09] Philip Eardley. Metering and Marking Behaviour of PCN-Nodes. Request for Comments 5670, RFC Editor, November 2009.

[FA08] Sally Floyd and Mark Allman. Comments on the Usefulness of Simple Best-Effort Traffic. Request for Comments RFC5290, RFC Editor, July 2008. (Individual submission to RFC Editor).

[GM02] Liang Guo and Ibrahim Matta. Scheduling Flows with Unknown Sizes: Approximate Analysis. *SIGMETRICS Perform. Eval. Rev.*, 30(1):276–277, June 2002.

[HJ18] Toke Høiland-Jørgensen. *Bufferbloat and Beyond; Removing Performance Barriers in Real-World Networks.* Phd thesis, Karlstad University, November 2018.

[HRPA17] David Andrew Hayes, David Ros, Andreas Petlund, and Iffat Ahmed. A Framework for Less than Best Effort Congestion Control with Soft Deadlines. IEEE, 2017.

[Joh17] Ingemar Johansson. Motivation to improved ECN handling in NR. Technical Report TSG-RAN WG2 #99 Tdoc R2-1709469, 3GPP, August 2017. http://www.3gpp.org/ftp/tsg_ran/WG2_RL2/TSGR2_99/Docs/R2-1709469.zip.

[KMT98] Frank P. Kelly, Aman K. Maulloo, and David K. H. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49(3):237–252, 1998.

[LVG07] S. Liu, M. Vojnovic, and D. Gunawardena. Competitive and Considerate Congestion Control for Bulk Data Transfers. In *15th Int'l Wkshp on Quality of Service*, pages 1–9. IEEE, June 2007.

[MANC09] Patrick Mulroy, Steve Appleby, Mike Nilsson, and Barry Crabtree. The Use of MulTCP for the Delivery of Equitable Quality Video. In *Proc. Int'l Packet Video Wkshp (PV'09)*. IEEE, May 2009.

[MH19] Jonathan Morton and Peter Heist. Lightweight Fair Queuing. Internet Draft draft-ietf-tsvwg-lightweight-fair-queuing-00, IETF, March 2019. (work in progress).

[MSSM12] G. R. Moktan, S. Siikavirta, M. Srel, and J. Manner. Favoring the short. In *Proc. IEEE INFOCOM Workshops*, pages 31–36, March 2012.

[MT19] Jonathan Morton and David Taeht. The Some Congestion Experienced ECN Codepoint. Internet Draft draft-morton-taht-sce-00, IETF, March 2019. (work in progress).

[RW13] D. Ros and M. Welzl. Less-than-Best-Effort Service: A Survey of End-to-End Approaches. *IEEE Communications Surveys Tutorials*, 15(2):898–908, February 2013.

[Sd02] Shanchieh Yang and G. de Veciana. Size-based adaptive bandwidth allocation: optimizing the average QoS for elastic flows. In *Proc. IEEE Conference on Computer Communications*, volume 2, pages 657–666, June 2002.

[SRC84] Jerome H. Saltzer, David P. Reed, and David D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277–288, November 1984.

[TE+15] Ing-Jyh Tsang (Ed.) et al. Deployment of RITE mechanisms in use-case trial testbeds report. Deliverable D3.3, RITE Eu FP7 Project 317700, November 2015.

[WSB19] Greg White, Karthik Sunderesan, and Bob Briscoe. Low Latency DOCSIS: Technology Overview. Internet Draft draft-white-tsvwg-lld, IETF, March 2019.

[ZTH04] Thomas Ziegler, Hung Tuan Tran, and Eduard Hasenleithner. Improving Perceived Web Performance by Size Based Congestion Control. In *Proc 3rd Int'l IFIP-TC6 Networking Conf.*, page 687, May 2004.

Document history

Version	Date	Author	Details of change
00A	20 Jul 2019	Bob Briscoe	First complete draft.
01	22 Jul 2019	Bob Briscoe	First Publication.