Transport Area Working Group B. Briscoe, Ed.
Internet-Draft CableLabs
Intended status: Informational K. De Schepper
Expires: January 9, 2020 Nokia Bell Labs
M. Bagnulo Braun
Universidad Carlos III de Madrid
G. White
CableLabs
July 8, 2019

Low Latency, Low Loss, Scalable Throughput (L4S) Internet Service: Architecture
draft-ietf-tsvwg-l4s-arch-04

Abstract

This document describes the L4S architecture for the provision of a new Internet service that could eventually replace best efforts for all traffic: Low Latency, Low Loss, Scalable throughput (L4S). It is becoming common for _all_ (or most) applications being run by a user at any one time to require low latency. However, the only solution the IETF can offer for ultra-low queuing delay is Diffserv, which only favours a minority of packets at the expense of others. In extensive testing the new L4S service keeps average queuing delay under a millisecond for _all_ applications even under very heavy load, without sacrificing utilization; and it keeps congestion loss to zero. It is becoming widely recognized that adding more access capacity gives diminishing returns, because latency is becoming the critical problem. Even with a high capacity broadband access, the reduced latency of L4S remarkably and consistently improves performance under load for applications such as interactive video, conversational video, voice, Web, gaming, instant messaging, remote desktop and cloud-based apps (even when all being used at once over the same access link). The insight is that the root cause of queuing delay is in TCP, not in the queue. By fixing the sending TCP (and other transports) queuing latency becomes so much better than today that operators will want to deploy the network part of L4S to enable new products and services. Further, the network part is simple to deploy - incrementally with zero-config. Both parts, sender and network, ensure coexistence with other legacy traffic. At the same time L4S solves the long-recognized problem with the future scalability of TCP throughput.

This document describes the L4S architecture, briefly describing the

different components and how the work together to provide the aforementioned enhanced Internet service.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

It is increasingly common for _all_ of a user's applications at any one time to require low delay: interactive Web, Web services, voice, conversational video, interactive video, interactive remote presence, instant messaging, online gaming, remote desktop, cloud-based applications and video-assisted remote control of machinery and industrial processes. In the last decade or so, much has been done to reduce propagation delay by placing caches or servers closer to users. However, queuing remains a major, albeit intermittent, component of latency. For instance spikes of hundreds of milliseconds are common. During a long-running flow, even with state-of-the-art active queue management (AQM), the base speed-of-light path delay roughly doubles. Low loss is also important because, for interactive applications, losses translate into even longer retransmission delays.

It has been demonstrated that, once access network bit rates reach levels now common in the developed world, increasing capacity offers diminishing returns if latency (delay) is not addressed. Differentiated services (Diffserv) offers Expedited Forwarding (EF [RFC3246]) for some packets at the expense of others, but this is not sufficient when all (or most) of a user's applications require low

latency.

Therefore, the goal is an Internet service with ultra-Low queueing Latency, ultra-Low Loss and Scalable throughput (L4S) - for _all_ traffic. A service for all traffic will need none of the configuration or management baggage (traffic policing, traffic contracts) associated with favouring some packets over others. This document describes the L4S architecture for achieving that goal.

It must be said that queuing delay only degrades performance infrequently [Hohlfeld14]. It only occurs when a large enough

capacity-seeking (e.g. TCP) flow is running alongside the user's traffic in the bottleneck link, which is typically in the access network. Or when the low latency application is itself a large capacity-seeking flow (e.g. interactive video). At these times, the performance improvement from L4S must be so remarkable that network operators will be motivated to deploy it.

Active Queue Management (AQM) is part of the solution to queuing under load. AQM improves performance for all traffic, but there is a limit to how much queuing delay can be reduced by solely changing the network; without addressing the root of the problem.

The root of the problem is the presence of standard TCP congestion control (Reno [RFC5681]) or compatible variants (e.g. TCP Cubic [RFC8312]). We shall call this family of congestion controls 'Classic' TCP. It has been demonstrated that if the sending host replaces Classic TCP with a 'Scalable' alternative, when a suitable AQM is deployed in the network the performance under load of all the above interactive applications can be stunningly improved. For instance, queuing delay under heavy load with the example DCTCP/DualQ solution cited below is roughly 1 millisecond (1 to 2 ms) at the 99th percentile without losing link utilization. This compares with 5 to 20 ms on _average_ with a Classic TCP and current state-of-the-art AQMs such as fq_CoDel [RFC8290] or PIE [RFC8033] and about 20-30 ms at the 99th percentile. Also, with a Classic TCP, 5 ms of queuing is usually only possible by losing some utilization.

It has been convincingly demonstrated [DCttH15] that it is possible to deploy such an L4S service alongside the existing best efforts service so that all of a user's applications can shift to it when their stack is updated. Access networks are typically designed with one link as the bottleneck for each site (which might be a home, small enterprise or mobile device), so deployment at a single network node should give nearly all the benefit. The L4S approach also requires component mechanisms at the endpoints to fulfill its goal. This document presents the L4S architecture, by describing the different components and how they interact to provide the scalable low-latency, low-loss, Internet service.

2. L4S Architecture Overview

There are three main components to the L4S architecture (illustrated in Figure 1):

1) Network: L4S traffic needs to be isolated from the queuing latency of Classic traffic. However, the two should be able to

freely share a common pool of capacity. This is because there is no way to predict how many flows at any one time might use each

service and capacity in access networks is too scarce to partition into two. The Dual Queue Coupled AQM [I-D.ietf-tsvwg-aqm-dualq-coupled] was developed as a minimal complexity solution to this problem. The two queues appear to be separated by a 'semi-permeable' membrane that partitions latency but not bandwidth (explained later).

Per-flow queuing such as in [RFC8290] could be used (see Section 4), but it partitions both latency and bandwidth between every end-to-end flow. So it is rather overkill, which brings disadvantages (see Section 5.2), not least that large number of queues are needed when two are sufficient.

2) Protocol: A host needs to distinguish L4S and Classic packets with an identifier so that the network can classify them into their separate treatments. [I-D.ietf-tsvwg-ecn-l4s-id] considers various alternative identifiers, and concludes that all alternatives involve compromises, but the ECT(1) and CE codepoints of the ECN field represent a workable solution.

3) Host: Scalable congestion controls already exist. They solve the scaling problem with TCP that was first pointed out in [RFC3649]. The one used most widely (in controlled environments) is Data Center TCP (DCTCP [RFC8257]), which has been implemented and deployed in Windows Server Editions (since 2012), in Linux and in FreeBSD. Although DCTCP as-is 'works' well over the public Internet, most implementations lack certain safety features that will be necessary once it is used outside controlled environments like data centres (see later). A similar scalable congestion control will also need to be transplanted into protocols other than TCP (QUIC, SCTP, RTP/RTCP, RMCAT, etc.) Indeed, between the present document being drafted and published, the following scalable congestion controls were implemented: TCP Prague [PragueLinux], QUIC Prague and an L4S variant of the RMCAT SCReAM controller [RFC8298].

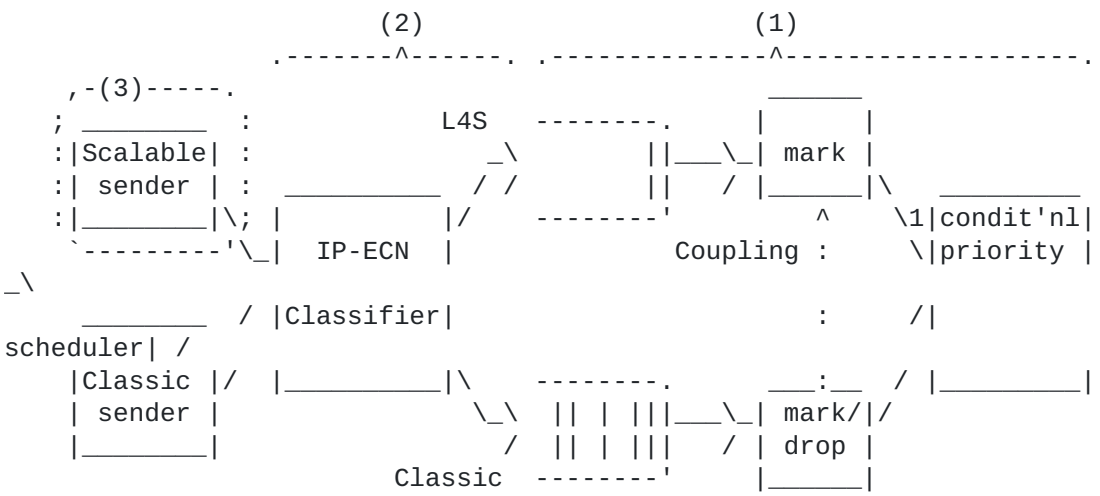


Figure 1: Components of an L4S Solution: 1) Isolation in separate network queues; 2) Packet Identification Protocol; and 3) Scalable Sending Host

3. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance. COMMENT: Since this will be an information document, This should be removed.

Classic service: The 'Classic' service is intended for all the congestion control behaviours that currently co-exist with TCP Reno (e.g. TCP Cubic, Compound, SCTP, etc).

Low-Latency, Low-Loss and Scalable (L4S) service: The 'L4S' service is intended for traffic from scalable TCP algorithms such as Data Center TCP. But it is also more general--it will allow a set of congestion controls with similar scaling properties to DCTCP (e.g. Relentless [Mathis09]) to evolve.

Both Classic and L4S services can cope with a proportion of unresponsive or less-responsive traffic as well (e.g. DNS, VoIP, etc).

Scalable Congestion Control: A congestion control where the packet flow rate per round trip (the window) is inversely proportional to the level (probability) of congestion signals. Then, as flow rate scales, the number of congestion signals per round trip remains

invariant, maintaining the same degree of control. For instance,

DCTCP averages 2 congestion signals per round-trip whatever the flow rate.

Classic Congestion Control: A congestion control with a flow rate that can co-exist with standard TCP Reno [RFC5681] without starvation. With Classic congestion controls, as capacity increases enabling higher flow rates, the number of round trips between congestion signals (losses or ECN marks) rises in proportion to the flow rate. So control of queuing and/or utilization becomes very slack. For instance, with 1500 B packets and an RTT of 18 ms, as TCP Reno flow rate increases from 2 to 100 Mb/s the number of round trips between congestion signals rises proportionately, from 2 to 100.

The default congestion control in Linux (TCP Cubic) is Reno-compatible for most Internet access scenarios expected for some years. For instance, with a typical domestic round-trip time (RTT) of 18ms, TCP Cubic only switches out of Reno-compatibility mode once the flow rate approaches 1 Gb/s. For a typical data centre RTT of 1 ms, the switch-over point is theoretically 1.3 Tb/s. However, with a less common transcontinental RTT of 100 ms, it only remains Reno-compatible up to 13 Mb/s. All examples assume 1,500 B packets.

Classic ECN: The original proposed standard Explicit Congestion Notification (ECN) protocol [RFC3168], which requires ECN signals to be treated the same as drops, both when generated in the network and when responded to by the sender.

Site: A home, mobile device, small enterprise or campus, where the network bottleneck is typically the access link to the site. Not all network arrangements fit this model but it is a useful, widely applicable generalisation.

4. L4S Architecture Components

The L4S architecture is composed of the following elements.

Protocols:The L4S architecture encompasses the two identifier changes (an unassignment and an assignment) and optional further identifiers:

a. An essential aspect of a scalable congestion control is the use of explicit congestion signals rather than losses, because the signals need to be sent immediately and frequently. 'Classic' ECN [RFC3168] requires an ECN signal to be treated the same as a

drop, both when it is generated in the network and when it is responded to by hosts. L4S needs networks and hosts to support a different meaning for ECN:

* much more frequent signals--too often to use drops;

* immediately tracking every fluctuation of the queue--too soon to commit to dropping packets.

So the standards track [RFC3168] has had to be updated to allow L4S packets to depart from the 'same as drop' constraint. [RFC8311] is a standards track update to relax specific requirements in RFC 3168 (and certain other standards track RFCs), which clears the way for the experimental changes proposed for L4S. [RFC8311] also reclassifies the original experimental assignment of the ECT(1) codepoint as an ECN nonce [RFC3540] as historic.

b. [I-D.ietf-tsvwg-ecn-l4s-id] recommends ECT(1) is used as the identifier to classify L4S packets into a separate treatment from Classic packets. This satisfies the requirements for identifying an alternative ECN treatment in [RFC4774].

The CE codepoint is used to indicate Congestion Experienced by both L4S and Classic treatments. This raises the concern that a Classic AQM earlier on the path might have marked some ECT(0) packets as CE. Then these packets will be erroneously classified into the L4S queue. [I-D.ietf-tsvwg-ecn-l4s-id] explains why 5 unlikely eventualities all have to coincide for this to have any detrimental effect, which even then would only involve a vanishingly small likelihood of a spurious retransmission.

c. A network operator might wish to include certain unresponsive, non-L4S traffic in the L4S queue if it is deemed to be smoothly enough paced and low enough rate not to build a queue. For instance, VoIP, low rate datagrams to sync online games, relatively low rate application-limited traffic, DNS, LDAP, etc. This traffic would need to be tagged with specific identifiers, e.g. a low latency Diffserv Codepoint such as Expedited Forwarding (EF [RFC3246]), Non-Queue-Building (NQB [I-D.white-tsvwg-nqb]), or operator-specific identifiers.

Network components: The L4S architecture encompasses either dual-queue or per-flow queue solutions:

a. The Dual Queue Coupled AQM has been specified as generically as possible [I-D.ietf-tsvwg-aqm-dualq-coupled] as a 'semi-permeable' membrane without specifying the particular AQMs to use in the two queues. Informational appendices of the draft are provided for pseudocode examples of different possible AQM approaches. The

aim is for designers to be free to implement diverse ideas. So the brief normative body of the draft only specifies the minimum

constraints an AQM needs to comply with to ensure that the L4S and Classic services will coexist. The core idea is the tension between the scheduler's prioritization of L4S over Classic and the coupling from the Classic to the L4S AQM. The L4S AQM derives its level of ECN marking from the maximum of the congestion levels in both queues. So L4S flows leave enough space between their packets for Classic flows, as if they were all the same type of TCP, all sharing one FIFO queue.

Initially a zero-config variant of RED called Curvy RED was implemented, tested and documented. Then, a variant of PIE called DualPI2 (pronounced Dual PI Squared) [PI2] was implemented and found to perform better than Curvy RED over a wide range of conditions, so it was documented in another appendix of [I-D.ietf-tsvwg-aqm-dualq-coupled].

b. A scheduler with per-flow queues can be used for L4S. It would be simple to modify an existing design such as FQ-CoDel or FQ-PIE, although this has not been implemented and evaluated because the goal of the original proponents of L4S was to avoid per-flow scheduling.

The idea would be to implement two AQMs (Classic and Scalable) and switch each per-flow queue to use an instance of the appropriate AQM for the flow, based on the ECN codepoints of the packets. Flows of non-ECN or ECT(0) packets would use a Classic AQM such as CoDel or PIE, while flows of ECT(1) packets without any ECT(0) packets would use a simple shallow threshold AQM with immediate (unsmoothed) marking. The FQ scheduler might work as is, because it is likely that L4S flows would be continually categorized as 'new' flows. However, this presumption has not been tested under a wide range of conditions. A variant of FQ-CoDel already exists that adapts to a shallower threshold AQM for ECN-capable packets.

Host mechanisms: The L4S architecture includes a number of mechanisms in the end host that we enumerate next:

a. Data Center TCP is the most widely used example of a scalable congestion control. It has been documented as an informational record of the protocol currently in use [RFC8257]. It will be necessary to define a number of safety features for a variant usable on the public Internet. A draft list of these, known as the TCP Prague requirements, has been drawn up (see Appendix A of [I-D.ietf-tsvwg-ecn-l4s-id]). The list also includes some optional performance improvements.

b. Transport protocols other than TCP use various congestion controls designed to be friendly with Classic TCP. Before they can use the L4S service, it will be necessary to implement scalable variants of each of these congestion control behaviours.

The following standards track RFCs currently define these protocols: ECN in TCP [RFC3168], in SCTP [RFC4960], in RTP [RFC6679], and in DCCP [RFC4340]. Not all are in widespread use,

but those that are will eventually need to be updated to allow a different congestion response, which they will have to indicate by using the ECT(1) codepoint. Scalable variants are under consideration for some new transport protocols that are themselves under development, e.g. QUIC [I-D.ietf-quic-transport] and certain real-time media congestion avoidance techniques (RMCAT) protocols.

c. ECN feedback is sufficient for L4S in some transport protocols (RTCP, DCCP) but not others:

* For the case of TCP, the feedback protocol for ECN embeds the assumption from Classic ECN that an ECN mark is the same as a drop, making it unusable for a scalable TCP. Therefore, the implementation of TCP receivers will have to be upgraded [RFC7560]. Work to standardize and implement more accurate ECN feedback for TCP (AccECN) is in progress [I-D.ietf-tcpm-accurate-ecn], [PragueLinux].

* ECN feedback is only roughly sketched in an appendix of the SCTP specification. A fuller specification has been proposed [I-D.stewart-tsvwg-sctpecn], which would need to be implemented and deployed before SCTCP could support L4S.

5. Rationale

5.1. Why These Primary Components?

Explicit congestion signalling (protocol): Explicit congestion signalling is a key part of the L4S approach. In contrast, use of

drop as a congestion signal creates a tension because drop is both

a useful signal (more would reduce delay) and an impairment (less would reduce delay):

* Explicit congestion signals can be used many times per round trip, to keep tight control, without any impairment. Under heavy load, even more explicit signals can be applied so the queue can be kept short whatever the load. Whereas state-of-the-art AQMs have to introduce very high packet drop at high load to keep the queue short. Further, when using ECN, TCP's sawtooth reduction can be smaller and therefore return to the

operating point more often, without worrying that this causes more signals (one at the top of each smaller sawtooth). The consequent smaller amplitude sawteeth fit between a very shallow marking threshold and an empty queue, so delay variation can be very low, without risk of under-utilization.

* Explicit congestion signals can be sent immediately to track fluctuations of the queue. L4S shifts smoothing from the network (which doesn't know the round trip times of all the flows) to the host (which knows its own round trip time). Previously, the network had to smooth to keep a worst-case round trip stable, delaying congestion signals by 100-200ms.

All the above makes it clear that explicit congestion signalling is only advantageous for latency if it does not have to be considered 'the same as' drop (as was required with Classic ECN [RFC3168]). Therefore, in a DualQ AQM, the L4S queue uses a new L4S variant of ECN that is not equivalent to drop [I-D.ietf-tsvwg-ecn-l4s-id], while the Classic queue uses either classic ECN [RFC3168] or drop, which are equivalent.

Before Classic ECN was standardized, there were various proposals to give an ECN mark a different meaning from drop. However, there was no particular reason to agree on any one of the alternative meanings, so 'the same as drop' was the only compromise that could be reached. RFC 3168 contains a statement that:

"An environment where all end nodes were ECN-Capable could allow new criteria to be developed for setting the CE codepoint, and new congestion control mechanisms for end-node reaction to CE packets. However, this is a research issue, and as such is not addressed in this document."

Latency isolation with coupled congestion notification (network):
Using just two queues is not essential to L4S (more would be possible), but it is the simplest way to isolate all the L4S traffic that keeps latency low from all the legacy Classic traffic that does not.

Similarly, coupling the congestion notification between the queues is not necessarily essential, but it is a clever and simple way to allow senders to determine their rate, packet-by-packet, rather than be overridden by a network scheduler. Because otherwise a network scheduler would have to inspect at least transport layer headers, and it would have to continually assign a rate to each flow without any easy way to understand application intent.

L4S packet identifier (protocol): Once there are at least two separate treatments in the network, hosts need an identifier at the IP layer to distinguish which treatment they intend to use.

Scalable congestion notification (host): A scalable congestion control keeps the signalling frequency high so that rate variations can be small when signalling is stable, and rate can track variations in available capacity as rapidly as possible otherwise.

Low loss: Latency is not the only concern of L4S. The 'Low Loss" part of the name denotes that L4S generally achieves zero congestion loss due to its use of ECN. Otherwise, loss would itself cause delay, particularly for short flows, due to retransmission delay [RFC2884].

Scalable throughput: The "Scalable throughput" part of the name denotes that the per-flow throughput of scalable congestion controls should scale indefinitely, avoiding the imminent scaling problems with TCP-Friendly congestion control algorithms [RFC3649]. It was known when TCP was first developed that it would not scale to high bandwidth-delay products (see footnote 6 in [TCP-CA]). Today, regular broadband bit-rates over WAN distances are already beyond the scaling range of `classic' TCP Reno. So `less unscalable' Cubic [RFC8312] and Compound [I-D.sridharan-tcpm-ctcp] variants of TCP have been successfully deployed. However, these are now approaching their scaling limits. For instance, at 800Mb/s with a 20ms round trip, Cubic induces a congestion signal only every 500 round trips or 10 seconds, which makes its dynamic control very sloppy. In contrast on average a scalable congestion control like DCTCP or TCP Prague induces 2 congestion signals per round trip, which remains invariant for any flow rate, keeping dynamic control very tight.

5.2. Why Not Alternative Approaches?

All the following approaches address some part of the same problem space as L4S. In each case, it is shown that L4S complements them or improves on them, rather than being a mutually exclusive alternative:

Diffserv: Diffserv addresses the problem of bandwidth apportionment for important traffic as well as queuing latency for delay-sensitive traffic. L4S solely addresses the problem of queuing latency (as well as loss and throughput scaling). Diffserv will still be necessary where important traffic requires priority (e.g. for commercial reasons, or for protection of critical infrastructure traffic) - see [I-D.briscoe-tsvwg-l4s-diffserv].

Nonetheless, if there are Diffserv classes for important traffic,

the L4S approach can provide low latency for _all_ traffic within each Diffserv class (including the case where there is only one Diffserv class).

Also, as already explained, Diffserv only works for a small subset of the traffic on a link. It is not applicable when all the applications in use at one time at a single site (home, small business or mobile device) require low latency. Also, because L4S is for all traffic, it needs none of the management baggage (traffic policing, traffic contracts) associated with favouring some packets over others. This baggage has held Diffserv back from widespread end-to-end deployment.

State-of-the-art AQMs: AQMs such as PIE and fq_CoDel give a significant reduction in queuing delay relative to no AQM at all. The L4S work is intended to complement these AQMs, and we definitely do not want to distract from the need to deploy them as widely as possible. Nonetheless, without addressing the large saw-toothing rate variations of Classic congestion controls, AQMs alone cannot reduce queuing delay too far without significantly reducing link utilization. The L4S approach resolves this tension by ensuring hosts can minimize the size of their sawteeth without appearing so aggressive to legacy flows that they starve them.

Per-flow queuing: Similarly per-flow queuing is not incompatible with the L4S approach. However, one queue for every flow can be thought of as overkill compared to the minimum of two queues for all traffic needed for the L4S approach. The overkill of per-flow queuing has side-effects:

A. fq makes high performance networking equipment costly (processing and memory) - in contrast dual queue code can be very simple;

B. fq requires packet inspection into the end-to-end transport layer, which doesn't sit well alongside encryption for privacy - in contrast the use of ECN as the classifier for L4S requires no deeper inspection than the IP layer;

C. fq isolates the queuing of each flow from the others but not from itself so existing FQ implementations still needs to have support for scalable congestion control added.

It might seem that self-inflicted queuing delay should not count, because if the delay wasn't in the network it would

just shift to the sender. However, modern adaptive applications, e.g. HTTP/2 [RFC7540] or the interactive media applications described in Section 6, can keep low latency

objects at the front of their local send queue by shuffling priorities of other objects dependent on the progress of other transfers. They cannot shuffle packets once they have released them into the network.

D. fq prevents any one flow from consuming more than 1/N of the capacity at any instant, where N is the number of flows. This is fine if all flows are elastic, but it does not sit well with a variable bit rate real-time multimedia flow, which requires wriggle room to sometimes take more and other times less than a 1/N share.

It might seem that an fq scheduler offers the benefit that it prevents individual flows from hogging all the bandwidth. However, L4S has been deliberately designed so that policing of individual flows can be added as a policy choice, rather than requiring one specific policy choice as the mechanism itself. A scheduler (like fq) has to decide packet-by-packet which flow to schedule without knowing application intent. Whereas a separate policing function can be configured less strictly, so that senders can still control the instantaneous rate of each flow dependent on the needs of each application (e.g. variable rate video), giving more wriggle-room before a flow is deemed non-compliant. Also policing of queuing and of flow-rates can be applied independently.

Alternative Back-off ECN (ABE): Yet again, L4S is not an alternative to ABE but a complement that introduces much lower queuing delay. ABE [RFC8511] alters the host behaviour in response to ECN marking to utilize a link better and give ECN flows faster throughput. It uses ECT(0) and assumes the network still treats ECN and drop the same. Therefore ABE exploits any lower queuing delay that AQMs can provide. But as explained above, AQMs still cannot reduce queuing delay too far without losing link utilization (to allow for other, non-ABE, flows).

BBRv1: v1 of Bottleneck Bandwidth and Round-trip propagation time (BBR [I-D.cardwell-iccrg-bbr-congestion-control]) controls queuing delay end-to-end without needing any special logic in the network, such as an AQM - so it works pretty-much on any path. Setting some problems with capacity sharing aside, queuing delay is good with BBRv1, but perhaps not quite as low as with state-of-the-art AQMs such as PIE or fq_CoDel, and certainly nowhere near as low as

with L4S. Queuing delay is also not consistently low, due to its regular bandwidth probes and the aggressive flow start-up phase.

L4S is a complement to BBRv1. Indeed BBRv2 (not released at the time of writing) is likely to use L4S ECN and a TCP-Prague-like

behaviour if it discovers a compatible path. Otherwise it will use an evolution of BBRv1.

6. Applicability

6.1. Applications

A transport layer that solves the current latency issues will provide new service, product and application opportunities.

With the L4S approach, the following existing applications will immediately experience significantly better quality of experience under load:

- Gaming;
- VoIP;
- Video conferencing;
- Web browsing;
- (Adaptive) video streaming;
- Instant messaging.

The significantly lower queuing latency also enables some interactive application functions to be offloaded to the cloud that would hardly even be usable today:

- Cloud based interactive video;
- Cloud based virtual and augmented reality.

The above two applications have been successfully demonstrated with L4S, both running together over a 40 Mb/s broadband access link loaded up with the numerous other latency sensitive applications in the previous list as well as numerous downloads - all sharing the same bottleneck queue simultaneously [L4Sdemo16]. For the former, a panoramic video of a football stadium could be swiped and pinched so that, on the fly, a proxy in the cloud could generate a sub-window of the match video under the finger-gesture control of each user. For the latter, a virtual reality headset displayed a viewport taken from a 360 degree camera in a racing car. The user's head movements controlled the viewport extracted by a cloud-based proxy. In both cases, with 7 ms end-to-end base delay, the additional queuing delay of roughly 1 ms was so low that it seemed the video was generated locally.

Using a swiping finger gesture or head movement to pan a video are extremely latency-demanding actions--far more demanding than VoIP. Because human vision can detect extremely low delays of the order of single milliseconds when delay is translated into a visual lag between a video and a reference point (the finger or the orientation of the head sensed by the balance system in the inner ear --- the vestibular system).

Without the low queuing delay of L4S, cloud-based applications like these would not be credible without significantly more access bandwidth (to deliver all possible video that might be viewed) and more local processing, which would increase the weight and power consumption of head-mounted displays. When all interactive processing can be done in the cloud, only the data to be rendered for the end user needs to be sent.

Other low latency high bandwidth applications such as:

- Interactive remote presence;
- Video-assisted remote control of machinery or industrial processes.

are not credible at all without very low queuing delay. No amount of extra access bandwidth or local processing can make up for lost time.

6.2. Use Cases

The following use-cases for L4S are being considered by various interested parties:

- Where the bottleneck is one of various types of access network: DSL, cable, mobile, satellite
  - Radio links (cellular, WiFi, satellite) that are distant from the source are particularly challenging. The radio link capacity can vary rapidly by orders of magnitude, so it is often desirable to hold a buffer to utilise sudden increases of capacity;
  - cellular networks are further complicated by a perceived need to buffer in order to make hand-overs imperceptible;
  - Satellite networks generally have a very large base RTT, so even with minimal queuing, overall delay can never be extremely low;

  * Nonetheless, it is certainly desirable not to hold a buffer purely because of the sawteeth of Classic TCP, when it is more than is needed for all the above reasons.

- Private networks of heterogeneous data centres, where there is no single administrator that can arrange for all the simultaneous changes to senders, receivers and network needed to deploy DCTCP:

  * a set of private data centres interconnected over a wide area with separate administrations, but within the same company

  * a set of data centres operated by separate companies interconnected by a community of interest network (e.g. for the finance sector)

  * multi-tenant (cloud) data centres where tenants choose their operating system stack (Infrastructure as a Service - IaaS)

- Different types of transport (or application) congestion control:

  * elastic (TCP/SCTP);

  * real-time (RTP, RMCAT);

  * query (DNS/LDAP).

- Where low delay quality of service is required, but without inspecting or intervening above the IP layer [I-D.smith-encrypted-traffic-management]:

  * mobile and other networks have tended to inspect higher layers in order to guess application QoS requirements. However, with growing demand for support of privacy and encryption, L4S offers an alternative. There is no need to select which traffic to favour for queuing, when L4S gives favourable queuing to all traffic.

- If queuing delay is minimized, applications with a fixed delay budget can communicate over longer distances, or via a longer chain of service functions [RFC7665] or onion routers.

6.3. Deployment Considerations

The DualQ is, in itself, an incremental deployment framework for L4S AQMs so that L4S traffic can coexist with existing Classic "TCP-friendly" traffic. Section 6.3.1 explains why only deploying a DualQ AQM [I-D.ietf-tsvwg-aqm-dualq-coupled] in one node at each end of the access link will realize nearly all the benefit of L4S.

L4S involves both end systems and the network, so Section 6.3.2 suggests some typical sequences to deploy each part, and why there will be an immediate and significant benefit after deploying just one part.

If an ECN-enabled DualQ AQM has not been deployed at a bottleneck, an L4S flow is required to include a fall-back strategy to Classic behaviour. Section 6.3.3 describes how an L4S flow detects this, and how to minimize the effect of false negative detection.

6.3.1. Deployment Topology

DualQ AQMs will not have to be deployed throughout the Internet before L4S will work for anyone. Operators of public Internet access networks typically design their networks so that the bottleneck will nearly always occur at one known (logical) link. This confines the cost of queue management technology to one place.

The case of mesh networks is different and will be discussed later. But the known bottleneck case is generally true for Internet access to all sorts of different 'sites', where the word 'site' includes home networks, small-to-medium sized campus or enterprise networks and even cellular devices (Figure 2). Also, this known-bottleneck case tends to be applicable whatever the access link technology; whether xDSL, cable, cellular, line-of-sight wireless or satellite.

Therefore, the full benefit of the L4S service should be available in the downstream direction when the DualQ AQM is deployed at the ingress to this bottleneck link (or links for multihomed sites). And similarly, the full upstream service will be available once the DualQ is deployed at the upstream ingress.

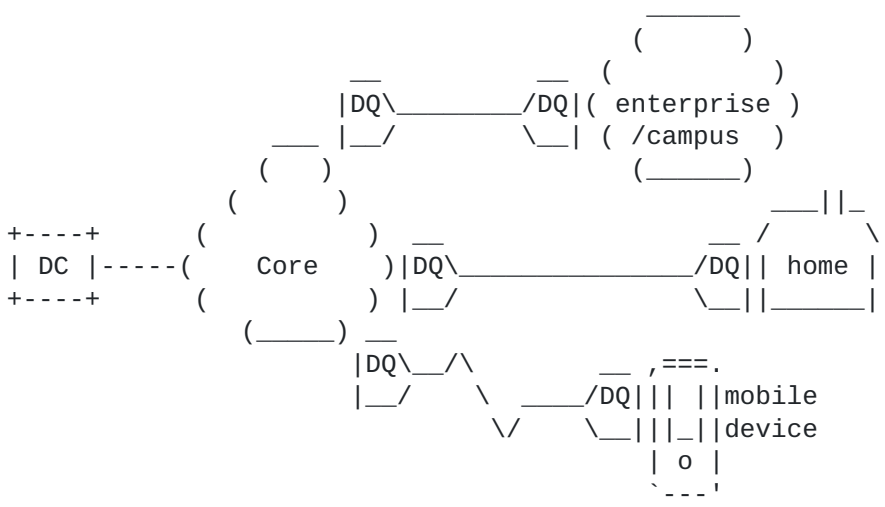


Figure 2: Likely location of DualQ (DQ) Deployments in common access topologies

Deployment in mesh topologies depends on how over-booked the core is.

If the core is non-blocking, or at least generously provisioned so that the edges are nearly always the bottlenecks, it would only be necessary to deploy the DualQ AQM at the edge bottlenecks. For example, some data-centre networks are designed with the bottleneck in the hypervisor or host NICs, while others bottleneck at the top-of-rack switch (both the output ports facing hosts and those facing the core).

The DualQ would eventually also need to be deployed at any other persistent bottlenecks such as network interconnections, e.g. some public Internet exchange points and the ingress and egress to WAN links interconnecting data-centres.

6.3.2. Deployment Sequences

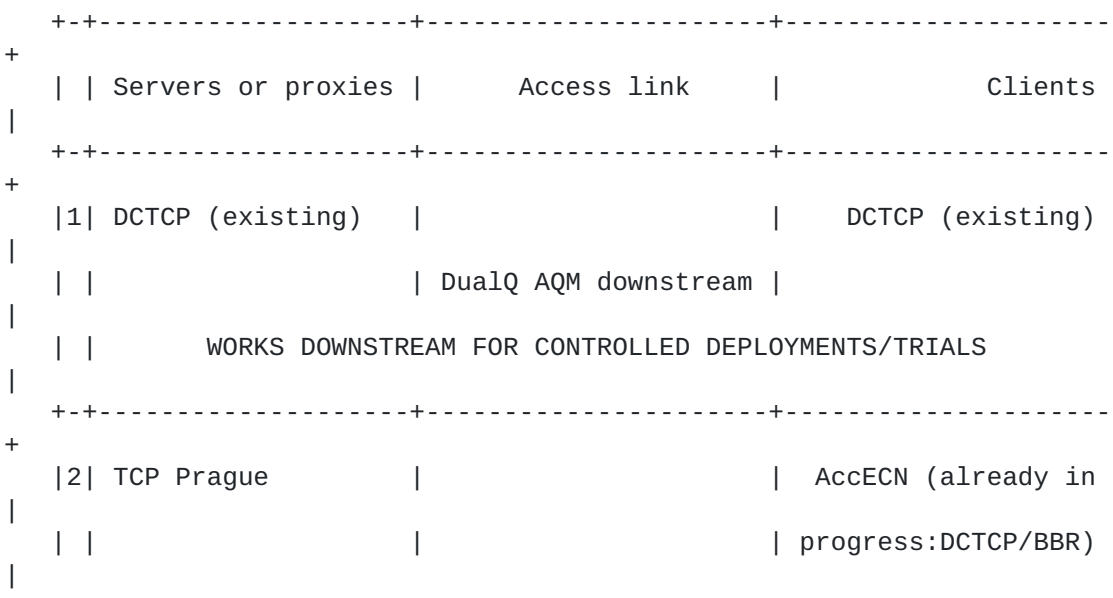
For any one L4S flow to work, it requires 3 parts to have been deployed. This was the same deployment problem that ECN faced [RFC8170] so we have learned from this.

Firstly, L4S deployment exploits the fact that DCTCP already exists on many Internet hosts (Windows, FreeBSD and Linux); both servers and clients. Therefore, just deploying DualQ AQM at a network bottleneck immediately gives a working deployment of all the L4S parts. DCTCP needs some safety concerns to be fixed for general use over the public Internet (see Section 2.3 of [I-D.ietf-tsvwg-ecn-l4s-id]), but

DCTCP is not on by default, so these issues can be managed within controlled deployments or controlled trials.

Secondly, the performance improvement with L4S is so significant that it enables new interactive services and products that were not previously possible. It is much easier for companies to initiate new work on deployment if there is budget for a new product trial. If, in contrast, there were only an incremental performance improvement (as with Classic ECN), spending on deployment tends to be much harder to justify.

Thirdly, the L4S identifier is defined so that initially network operators can enable L4S exclusively for certain customers or certain applications. But this is carefully defined so that it does not compromise future evolution towards L4S as an Internet-wide service. This is because the L4S identifier is defined not only as the end-to-end ECN field, but it can also optionally be combined with any other packet header or some status of a customer or their access link [I-D.ietf-tsvwg-ecn-l4s-id]. Operators could do this anyway, even if it were not blessed by the IETF. However, it is best for the IETF to specify that they must use their own local identifier in combination with the IETF's identifier. Then, if an operator enables the optional local-use approach, they only have to remove this extra rule to make the service work Internet-wide - it will already traverse middleboxes, peerings, etc.



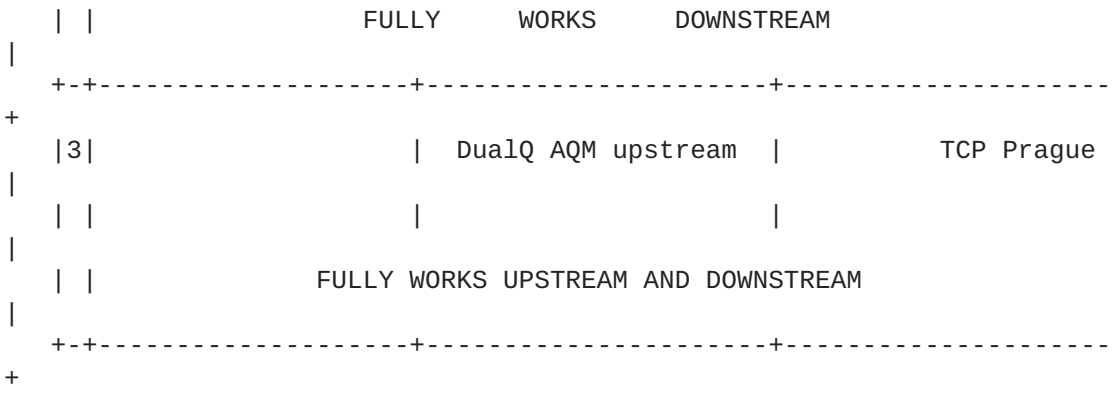


Figure 3: Example L4S Deployment Sequences

Figure 3 illustrates some example sequences in which the parts of L4S might be deployed. It consists of the following stages:

1. Here, the immediate benefit of a single AQM deployment can be seen, but limited to a controlled trial or controlled deployment.
   In this example downstream deployment is first, but in other scenarios the upstream might be deployed first. If no AQM at all was previously deployed for the downstream access, the DualQ AQM greatly improves the Classic service (as well as adding the L4S service). If an AQM was already deployed, the Classic service will be unchanged (and L4S will still be added).

2. In this stage, the name 'TCP Prague' is used to represent a variant of DCTCP that is safe to use in a production environment.
   If the application is primarily unidirectional, 'TCP Prague' at one end will provide all the benefit needed. Accurate ECN feedback (AccECN) [I-D.ietf-tcpm-accurate-ecn] is needed at the other end, but it is a generic ECN feedback facility that is already planned to be deployed for other purposes, e.g. DCTCP, BBR [I-D.cardwell-iccrg-bbr-congestion-control]. The two ends can be deployed in either order, because TCP Prague only enables itself if it has negotiated the use of AccECN feedback with the other end during the connection handshake. Thus, deployment of TCP Prague on a server enables L4S trials to move to a production service in one direction, wherever AccECN is deployed at the other end. This stage might be further motivated by the performance improvements of TCP Prague relative to DCTCP (see Appendix A.2 of [I-D.ietf-tsvwg-ecn-l4s-id]).

3. This is a two-move stage to enable L4S upstream. The DualQ or TCP Prague can be deployed in either order as already explained. To motivate the first of two independent moves, the deferred benefit of enabling new services after the second move has to be worth it to cover the first mover's investment risk. As explained already, the potential for new interactive services provides this motivation. The DualQ AQM also greatly improves the upstream Classic service, assuming no other AQM has already been deployed.

Note that other deployment sequences might occur. For instance: the upstream might be deployed first; a non-TCP protocol might be used end-to-end, e.g. QUIC, RMCAT; a body such as the 3GPP might require L4S to be implemented in 5G user equipment, or other random acts of kindness.

6.3.3. L4S Flow but Non-L4S Bottleneck

If L4S is enabled between two hosts but there is no L4S AQM at the bottleneck, any drop from the bottleneck will trigger the L4S sender to fall back to a classic ('TCP-Friendly') behaviour (see Appendix A.1.3 of [I-D.ietf-tsvwg-ecn-l4s-id]).

Unfortunately, as well as protecting legacy traffic, this rule degrades the L4S service whenever there is a loss, even if the loss was not from a non-DualQ bottleneck (false negative). And unfortunately, prevalent drop can be due to other causes, e.g.:

- congestion loss at other transient bottlenecks, e.g. due to bursts in shallower queues;
- transmission errors, e.g. due to electrical interference;
- rate policing.

Three complementary approaches are in progress to address this issue, but they are all currently research:

- In TCP Prague, ignore certain losses deemed unlikely to be due to congestion (using some ideas from BBR [I-D.cardwell-iccrg-bbr-congestion-control] but with no need to ignore nearly all losses). This could mask any of the above types of loss (requires consensus on how to safely interoperate with drop-based congestion controls).
- A combination of RACK, reconfigured link retransmission and L4S could address transmission errors [UnorderedLTE], [I-D.ietf-tsvwg-ecn-l4s-id];
- Hybrid ECN/drop policers (see Section 8.3).

L4S deployment scenarios that minimize these issues (e.g. over wireline networks) can proceed in parallel to this research, in the expectation that research success could continually widen L4S applicability.

Classic ECN support is starting to materialize on the Internet as an increased level of CE marking. Given some of this Classic ECN might be due to single-queue ECN deployment, an L4S sender will have to fall back to a classic ('TCP-Friendly') behaviour if it detects that ECN marking is accompanied by greater queuing delay or greater delay variation than would be expected with L4S (see Appendix A.1.4 of [I-D.ietf-tsvwg-ecn-l4s-id]). It is hard to detect whether this is all due to the addition of support for ECN in the Linux implementation of FQ-CoDel, which would not require fall-back to Classic behaviour, because FQ inherently forces the throughput of each flow to be equal irrespective of its aggressiveness.

6.3.4. Other Potential Deployment Issues

An L4S AQM uses the ECN field to signal congestion. So, in common with Classic ECN, if the AQM is within a tunnel or at a lower layer, correct functioning of ECN signalling requires correct propagation of the ECN field up the layers [RFC6040], [I-D.ietf-tsvwg-ecn-encap-guidelines].

7. IANA Considerations

This specification contains no IANA considerations.

8. Security Considerations

8.1. Traffic (Non-)Policing

Because the L4S service can serve all traffic that is using the capacity of a link, it should not be necessary to police access to the L4S service. In contrast, Diffserv only works if some packets get less favourable treatment than others. So Diffserv has to use traffic policers to limit how much traffic can be favoured. In turn, traffic policers require traffic contracts between users and networks as well as pairwise between networks. Because L4S will lack all this management complexity, it is more likely to work end-to-end.

During early deployment (and perhaps always), some networks will not offer the L4S service. These networks do not need to police or re-mark L4S traffic - they just forward it unchanged as best efforts traffic, as they already forward traffic with ECT(1) today. At a bottleneck, such networks will introduce some queuing and dropping. When a scalable congestion control detects a drop it will have to respond as if it is a Classic congestion control (as required in Section 2.3 of [I-D.ietf-tsvwg-ecn-l4s-id]). This will ensure safe interworking with other traffic at the 'legacy' bottleneck, but it will degrade the L4S service to no better (but never worse) than classic best efforts, whenever a legacy (non-L4S) bottleneck is encountered on a path.

Certain network operators might choose to restrict access to the L4S class, perhaps only to selected premium customers as a value-added service. Their packet classifier (item 2 in Figure 1) could identify such customers against some other field (e.g. source address range) as well as ECN. If only the ECN L4S identifier matched, but not the source address (say), the classifier could direct these packets (from non-premium customers) into the Classic queue. Clearly explaining how operators can use an additional local classifiers (see

[I-D.ietf-tsvwg-ecn-l4s-id]) is intended to remove any tendency to bleach the L4S identifier. Then at least the L4S ECN identifier will

be more likely to survive end-to-end even though the service may not be supported at every hop. Such arrangements would only require simple registered/not-registered packet classification, rather than the managed, application-specific traffic policing against customer-specific traffic contracts that Diffserv uses.

8.2. 'Latency Friendliness'

The L4S service does rely on self-constraint - not in terms of limiting rate, but in terms of limiting latency (burstiness). It is hoped that self-interest and standardisation of dynamic behaviour (cf. TCP slow-start) will be sufficient to prevent transports from sending excessive bursts of L4S traffic, given the application's own latency will suffer most from such behaviour.

Whether burst policing becomes necessary remains to be seen. Without it, there will be potential for attacks on the low latency of the L4S service. However it may only be necessary to apply such policing reactively, e.g. punitively targeted at any deployments of new bursty malware.

A per-flow (5-tuple) queue protection function [I-D.briscoe-docsis-q-protection] has been developed for the low latency queue in DOCSIS, which has adopted the DualQ L4S architecture. It protects the low latency service from any queue-building flows that accidentally or maliciously classify themselves into the low latency queue. It is designed to score flows based solely on their contribution to queuing (not flow rate in itself). Then, if the shared low latency queue is at risk of exceeding a threshold, the function redirects enough packets of the highest scoring flow(s) into the Classic queue to preserve low latency.

Such a queue protection function is not considered a necessary part of the L4S architecture, which works without it (in a similar way to how the Internet works without per-flow rate policing). Indeed, under normal circumstances, DOCSIS queue protection does not intervene, and if operators find it is not necessary they can disable it. Part of the L4S experiment will be to see whether such a function is necessary.

8.3. Interaction between Rate Policing and L4S

As mentioned in Section 5.2, L4S should remove the need for low latency Diffserv classes. However, those Diffserv classes that give certain applications or users priority over capacity, would still be applicable in certain scenarios (e.g. corporate networks). Then, within such Diffserv classes, L4S would often be applicable to give traffic low latency and low loss as well. Within such a Diffserv

class, the bandwidth available to a user or application is often limited by a rate policer. Similarly, in the default Diffserv class, rate policers are used to partition shared capacity.

A classic rate policer drops any packets exceeding a set rate, usually also giving a burst allowance (variants exist where the policer re-marks non-compliant traffic to a discard-eligible Diffserv codepoint, so they may be dropped elsewhere during contention). Whenever L4S traffic encounters one of these rate policers, it will experience drops and the source has to fall back to a Classic congestion control, thus losing the benefits of L4S. So, in networks that already use rate policers and plan to deploy L4S, it will be preferable to redesign these rate policers to be more friendly to the L4S service.

This is currently a research area. It might be achieved by setting a threshold where ECN marking is introduced, such that it is just under the policed rate or just under the burst allowance where drop is introduced. This could be applied to various types of policer, e.g. [RFC2697], [RFC2698] or the 'local' (non-ConEx) variant of the ConEx congestion policer [I-D.briscoe-conex-policing]. It might also be possible to design scalable congestion controls to respond less catastrophically to loss that has not been preceded by a period of increasing delay.

The design of L4S-friendly rate policers will require a separate dedicated document. For further discussion of the interaction between L4S and Diffserv, see [I-D.briscoe-tsvwg-l4s-diffserv].

8.4. ECN Integrity

Receiving hosts can fool a sender into downloading faster by suppressing feedback of ECN marks (or of losses if retransmissions are not necessary or available otherwise). Various ways to protect TCP feedback integrity have been developed. For instance:

- The sender can test the integrity of the receiver's feedback by occasionally setting the IP-ECN field to the congestion experienced (CE) codepoint, which is normally only set by a congested link. Then the sender can test whether the receiver's feedback faithfully reports what it expects (see 2nd para of Section 20.2 of [RFC3168]).

- A network can enforce a congestion response to its ECN markings (or packet losses) by auditing congestion exposure (ConEx) [RFC7713].

- The TCP authentication option (TCP-AO [RFC5925]) can be used to detect tampering with TCP congestion feedback.

- The ECN Nonce [RFC3540] was proposed to detect tampering with congestion feedback, but it has been reclassified as historic [RFC8311].

Appendix C.1 of [I-D.ietf-tsvwg-ecn-l4s-id] gives more details of these techniques including their applicability and pros and cons.

9. Acknowledgements

Thanks to Richard Scheffenegger, Wes Eddy, Karen Nielsen and David Black for their useful review comments.

Bob Briscoe and Koen De Schepper were part-funded by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700). Bob Briscoe was also part-funded by the Research Council of Norway through the TimeIn project. The views expressed here are solely those of the authors.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

10.2. Informative References

[DCttH15] De Schepper, K., Bondarenko, O., Briscoe, B., and I. Tsang, "`Data Centre to the Home': Ultra-Low Latency for All", RITE project Technical Report , 2015, <http://riteproject.eu/publications/>.

[Hohlfeld14] Hohlfeld , O., Pujol, E., Ciucu, F., Feldmann, A., and P. Barford, "A QoE Perspective on Sizing Network Buffers", Proc. ACM Internet Measurement Conf (IMC'14) hmm, November 2014.

[I-D.briscoe-conex-policing] Briscoe, B., "Network Performance Isolation using Congestion Policing", draft-briscoe-conex-policing-01 (work in progress), February 2014.

[I-D.briscoe-docsis-q-protection]
Briscoe, B. and G. White, "Queue Protection to Preserve Low Latency", draft-briscoe-docsis-q-protection-00 (work in progress), July 2019.

[I-D.briscoe-tsvwg-l4s-diffserv]
Briscoe, B., "Interactions between Low Latency, Low Loss, Scalable Throughput (L4S) and Differentiated Services", draft-briscoe-tsvwg-l4s-diffserv-02 (work in progress), November 2018.

[I-D.cardwell-iccrg-bbr-congestion-control]
Cardwell, N., Cheng, Y., Yeganeh, S., and V. Jacobson, "BBR Congestion Control", draft-cardwell-iccrg-bbr-congestion-control-00 (work in progress), July 2017.

[I-D.ietf-quic-transport]
Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-20 (work in progress), April 2019.

[I-D.ietf-tcpm-accurate-ecn]
Briscoe, B., Kuehlewind, M., and R. Scheffenegger, "More Accurate ECN Feedback in TCP", draft-ietf-tcpm-accurate-ecn-08 (work in progress), March 2019.

[I-D.ietf-tcpm-generalized-ecn]
Bagnulo, M. and B. Briscoe, "ECN++: Adding Explicit Congestion Notification (ECN) to TCP Control Packets", draft-ietf-tcpm-generalized-ecn-03 (work in progress), October 2018.

[I-D.ietf-tsvwg-aqm-dualq-coupled]
Schepper, K., Briscoe, B., and G. White, "DualQ Coupled AQMs for Low Latency, Low Loss and Scalable Throughput (L4S)", draft-ietf-tsvwg-aqm-dualq-coupled-09 (work in progress), July 2019.

[I-D.ietf-tsvwg-ecn-encap-guidelines]
Briscoe, B., Kaippallimalil, J., and P. Thaler, "Guidelines for Adding Congestion Notification to Protocols that Encapsulate IP", draft-ietf-tsvwg-ecn-encap-guidelines-13 (work in progress), May 2019.

[I-D.ietf-tsvwg-ecn-l4s-id]
Schepper, K. and B. Briscoe, "Identifying Modified Explicit Congestion Notification (ECN) Semantics for Ultra-Low Queuing Delay (L4S)", draft-ietf-tsvwg-ecn-l4s-id-06 (work in progress), March 2019.

[I-D.smith-encrypted-traffic-management]
Smith, K., "Network management of encrypted traffic", draft-smith-encrypted-traffic-management-05 (work in progress), May 2016.

[I-D.sridharan-tcpm-ctcp]
Sridharan, M., Tan, K., Bansal, D., and D. Thaler, "Compound TCP: A New TCP Congestion Control for High-Speed and Long Distance Networks", draft-sridharan-tcpm-ctcp-02 (work in progress), November 2008.

[I-D.stewart-tsvwg-sctpecn]
Stewart, R., Tuexen, M., and X. Dong, "ECN for Stream Control Transmission Protocol (SCTP)", draft-stewart-tsvwg-sctpecn-05 (work in progress), January 2014.

[I-D.white-tsvwg-nqb]
White, G. and T. Fossati, "Identifying and Handling Non Queue Building Flows in a Bottleneck Link", draft-white-tsvwg-nqb-02 (work in progress), June 2019.

[L4Sdemo16]
Bondarenko, O., De Schepper, K., Tsang, I., and B. Briscoe, "orderedUltra-Low Delay for All: Live Experience, Live Analysis", Proc. MMSYS'16 pp33:1--33:4, May 2016, <http://dl.acm.org/citation.cfm?doid=2910017.2910633 (videos of demos: https://riteproject.eu/dctth/#1511dispatchwg)>.

[Mathis09]
Mathis, M., "Relentless Congestion Control", PFLDNeT'09 , May 2009, <https://www.gdt.id.au/~gdt/presentations/2010-07-06-questnet-tcp/reference-materials/papers/mathis-relentless-congestion-control.pdf>.

[NewCC_Proc]
Eggert, L., "Experimental Specification of New Congestion Control Algorithms", IETF Operational Note ion-tsv-alt-cc, July 2007.

[PI2] De Schepper, K., Bondarenko, O., Tsang, I., and B. Briscoe, "PI^2 : A Linearized AQM for both Classic and Scalable TCP", Proc. ACM CoNEXT 2016 pp.105-119, December 2016, <http://dl.acm.org/citation.cfm?doid=2999572.2999578>.

[PragueLinux] Briscoe, B., De Schepper, K., Albisser, O., Misund, J., Tilmans, O., Kuehlewind, M., and A. Ahmed, "Implementing the `TCP Prague' Requirements for Low Latency Low Loss Scalable Throughput (L4S)", Proc. Linux Netdev 0x13 , March 2019, <https://www.netdevconf.org/0x13/session.html?talk-tcp-prague-l4s>.

[RFC2697] Heinanen, J. and R. Guerin, "A Single Rate Three Color Marker", RFC 2697, DOI 10.17487/RFC2697, September 1999, <https://www.rfc-editor.org/info/rfc2697>.

[RFC2698] Heinanen, J. and R. Guerin, "A Two Rate Three Color Marker", RFC 2698, DOI 10.17487/RFC2698, September 1999, <https://www.rfc-editor.org/info/rfc2698>.

[RFC2884] Hadi Salim, J. and U. Ahmed, "Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks", RFC 2884, DOI 10.17487/RFC2884, July 2000, <https://www.rfc-editor.org/info/rfc2884>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <https://www.rfc-editor.org/info/rfc3168>.

[RFC3246] Davie, B., Charny, A., Bennet, J., Benson, K., Le Boudec, J., Courtney, W., Davari, S., Firoiu, V., and D. Stiliadis, "An Expedited Forwarding PHB (Per-Hop Behavior)", RFC 3246, DOI 10.17487/RFC3246, March 2002, <https://www.rfc-editor.org/info/rfc3246>.

[RFC3540] Spring, N., Wetherall, D., and D. Ely, "Robust Explicit Congestion Notification (ECN) Signaling with Nonces", RFC 3540, DOI 10.17487/RFC3540, June 2003, <https://www.rfc-editor.org/info/rfc3540>.

[RFC3649] Floyd, S., "HighSpeed TCP for Large Congestion Windows", RFC 3649, DOI 10.17487/RFC3649, December 2003, <https://www.rfc-editor.org/info/rfc3649>.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, DOI 10.17487/RFC4340, March 2006, <https://www.rfc-editor.org/info/rfc4340>.

[RFC4774] Floyd, S., "Specifying Alternate Semantics for the Explicit Congestion Notification (ECN) Field", BCP 124, RFC 4774, DOI 10.17487/RFC4774, November 2006, <https://www.rfc-editor.org/info/rfc4774>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <https://www.rfc-editor.org/info/rfc4960>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <https://www.rfc-editor.org/info/rfc5681>.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <https://www.rfc-editor.org/info/rfc5925>.

[RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <https://www.rfc-editor.org/info/rfc6040>.

[RFC6679] Westerlund, M., Johansson, I., Perkins, C., O'Hanlon, P., and K. Carlberg, "Explicit Congestion Notification (ECN) for RTP over UDP", RFC 6679, DOI 10.17487/RFC6679, August 2012, <https://www.rfc-editor.org/info/rfc6679>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <https://www.rfc-editor.org/info/rfc7540>.

[RFC7560] Kuehlewind, M., Ed., Scheffenegger, R., and B. Briscoe, "Problem Statement and Requirements for Increased Accuracy in Explicit Congestion Notification (ECN) Feedback", RFC 7560, DOI 10.17487/RFC7560, August 2015, <https://www.rfc-editor.org/info/rfc7560>.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <https://www.rfc-editor.org/info/rfc7665>.

[RFC7713] Mathis, M. and B. Briscoe, "Congestion Exposure (ConEx) Concepts, Abstract Mechanism, and Requirements", RFC 7713, DOI 10.17487/RFC7713, December 2015, <https://www.rfc-editor.org/info/rfc7713>.

[RFC8033] Pan, R., Natarajan, P., Baker, F., and G. White, "Proportional Integral Controller Enhanced (PIE): A Lightweight Control Scheme to Address the Bufferbloat Problem", RFC 8033, DOI 10.17487/RFC8033, February 2017, <https://www.rfc-editor.org/info/rfc8033>.

[RFC8170] Thaler, D., Ed., "Planning for Protocol Adoption and Subsequent Transitions", RFC 8170, DOI 10.17487/RFC8170, May 2017, <https://www.rfc-editor.org/info/rfc8170>.

[RFC8257] Bensley, S., Thaler, D., Balasubramanian, P., Eggert, L., and G. Judd, "Data Center TCP (DCTCP): TCP Congestion Control for Data Centers", RFC 8257, DOI 10.17487/RFC8257, October 2017, <https://www.rfc-editor.org/info/rfc8257>.

[RFC8290] Hoeiland-Joergensen, T., McKenney, P., Taht, D., Gettys, J., and E. Dumazet, "The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm", RFC 8290, DOI 10.17487/RFC8290, January 2018, <https://www.rfc-editor.org/info/rfc8290>.

[RFC8298] Johansson, I. and Z. Sarker, "Self-Clocked Rate Adaptation for Multimedia", RFC 8298, DOI 10.17487/RFC8298, December 2017, <https://www.rfc-editor.org/info/rfc8298>.

[RFC8311] Black, D., "Relaxing Restrictions on Explicit Congestion Notification (ECN) Experimentation", RFC 8311, DOI 10.17487/RFC8311, January 2018, <https://www.rfc-editor.org/info/rfc8311>.

[RFC8312] Rhee, I., Xu, L., Ha, S., Zimmermann, A., Eggert, L., and R. Scheffenegger, "CUBIC for Fast Long-Distance Networks", RFC 8312, DOI 10.17487/RFC8312, February 2018, <https://www.rfc-editor.org/info/rfc8312>.

[RFC8511] Khademi, N., Welzl, M., Armitage, G., and G. Fairhurst, "TCP Alternative Backoff with ECN (ABE)", RFC 8511, DOI 10.17487/RFC8511, December 2018, <https://www.rfc-editor.org/info/rfc8511>.

[TCP-CA] Jacobson, V. and M. Karels, "Congestion Avoidance and Control", Laurence Berkeley Labs Technical Report ,

November 1988, <http://ee.lbl.gov/papers/congavoid.pdf>.

[TCP-sub-mss-w]
Briscoe, B. and K. De Schepper, "Scaling TCP's Congestion Window for Small Round Trip Times", BT Technical Report TR-TUB8-2015-002, May 2015, <http://www.bobbriscoe.net/projects/latency/sub-mss-w.pdf>.

[UnorderedLTE]
Austrheim, M., "Implementing immediate forwarding for 4G in a network simulator", Masters Thesis, Uni Oslo , June 2019.

Appendix A. Standardization items

The following table includes all the items that will need to be standardized to provide a full L4S architecture.

The table is too wide for the ASCII draft format, so it has been split into two, with a common column of row index numbers on the left.

The columns in the second part of the table have the following meanings:

WG: The IETF WG most relevant to this requirement. The "tcpm/iccrg" combination refers to the procedure typically used for congestion control changes, where tcpm owns the approval decision, but uses the iccrg for expert review [NewCC_Proc];

TCP: Applicable to all forms of TCP congestion control;

DCTCP: Applicable to Data Center TCP as currently used (in controlled environments);

DCTCP bis: Applicable to an future Data Center TCP congestion control intended for controlled environments;

XXX Prague: Applicable to a Scalable variant of XXX (TCP/SCTP/RMCAT) congestion control.

Req #	Requirement	Reference
0	ARCHITECTURE	
1	L4S IDENTIFIER	[I-D.ietf-tsvwg-ecn-l4s-id]
2	DUAL QUEUE AQM	[I-D.ietf-tsvwg-aqm-dualq-coupled]
3	Suitable ECN Feedback	[I-D.ietf-tcpm-accurate-ecn], [I-D.stewart-tsvwg-sctpecn].
	SCALABLE TRANSPORT - SAFETY ADDITIONS	
4-1	Fall back to Reno/Cubic on loss	[I-D.ietf-tsvwg-ecn-l4s-id] S.2.3, [RFC8257]
4-2	Fall back to Reno/Cubic if classic ECN bottleneck detected	[I-D.ietf-tsvwg-ecn-l4s-id] S.2.3
4-3	Reduce RTT-dependence	[I-D.ietf-tsvwg-ecn-l4s-id] S.2.3
4-4	Scaling TCP's Congestion Window for Small Round Trip Times	[I-D.ietf-tsvwg-ecn-l4s-id] S.2.3, [TCP-sub-mss-w]
	SCALABLE TRANSPORT - PERFORMANCE	

		ENHANCEMENTS	
	5-1	Setting ECT in TCP Control Packets and Retransmissions	[I-D.ietf-tcpm-generalized-ecn]
	5-2	Faster-than-additive increase	[I-D.ietf-tsvwg-ecn-l4s-id] (Appx A.2.2)
	5-3	Faster Convergence at Flow Start	[I-D.ietf-tsvwg-ecn-l4s-id] (Appx A.2.2)

#	WG	TCP	DCTCP	DCTCP-bis	TCP Prague	SCTP Prague	RMCAT Prague
0	tsvwg	Y	Y	Y	Y	Y	Y
1	tsvwg			Y	Y	Y	Y
2	tsvwg	n/a	n/a	n/a	n/a	n/a	n/a
3	tcpm	Y	Y	Y	Y	n/a	n/a
4-1	tcpm		Y	Y	Y	Y	Y
4-2	tcpm/ iccrg?				Y	Y	?
4-3	tcpm/ iccrg?			Y	Y	Y	?
4-4	tcpm	Y	Y	Y	Y	Y	?
5-1	tcpm	Y	Y	Y	Y	n/a	n/a

```
   |     |        |     |       |           |        |        |
|
   | 5-2 | tcpm/  |     |       | Y         | Y      | Y      | ?
|
   |     | iccrg? |     |       |           |        |        |
|
   | 5-3 | tcpm/  |     |       | Y         | Y      | Y      | ?
|
   |     | iccrg? |     |       |           |        |        |
|
   +-----+--------+-----+-------+-----------+--------+--------+--------
+
```

Authors' Addresses

Bob Briscoe (editor)
CableLabs
UK

Email: ietf@bobbriscoe.net
URI: http://bobbriscoe.net/

Koen De Schepper
Nokia Bell Labs
Antwerp
Belgium

Email: koen.de_schepper@nokia.com
URI: https://www.bell-labs.com/usr/koen.de_schepper

Marcelo Bagnulo
Universidad Carlos III de Madrid
Av. Universidad 30
Leganes, Madrid 28911
Spain

Phone: 34 91 6249500
Email: marcelo@it.uc3m.es
URI: http://www.it.uc3m.es

Greg White
CableLabs
US

Email: G.White@CableLabs.com