Internet Engineering Task Force (IETF)
Request for Comments: 9331
Category: Experimental
ISSN: 2070-1721

K. De Schepper
Nokia Bell Labs
B. Briscoe, Ed.
Independent

January 2023

The Explicit Congestion Notification (ECN) Protocol for Low Latency, Low Loss, and Scalable Throughput (L4S)

Abstract

This specification defines the protocol to be used for a new network service called Low Latency, Low Loss, and Scalable throughput (L4S). L4S uses an Explicit Congestion Notification (ECN) scheme at the IP layer that is similar to the original (or 'Classic') ECN approach, except as specified within. L4S uses 'Scalable' congestion control, which induces much more frequent control signals from the network, and it responds to them with much more fine-grained adjustments so that very low (typically sub-millisecond on average) and consistently low queuing delay becomes possible for L4S traffic without compromising link utilization. Thus, even capacity-seeking (TCP-like) traffic can have high bandwidth and very low delay at the same time, even during periods of high traffic load.

The L4S identifier defined in this document distinguishes L4S from 'Classic' (e.g., TCP-Reno-friendly) traffic. Then, network bottlenecks can be incrementally modified to distinguish and isolate existing traffic that still follows the Classic behaviour, to prevent it from degrading the low queuing delay and low loss of L4S traffic. This Experimental specification defines the rules that L4S transports and network elements need to follow, with the intention that L4S flows neither harm each other's performance nor that of Classic traffic. It also suggests open questions to be investigated during experimentation. Examples of new Active Queue Management (AQM) marking algorithms and new transports (whether TCP-like or real time) are specified separately.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This Experimental specification defines the protocol to be used for a new network service called Low Latency, Low Loss, and Scalable throughput (L4S). L4S uses an Explicit Congestion Notification (ECN) scheme at the IP layer with the same set of codepoint transitions as the original (or 'Classic') ECN [RFC3168]. [RFC3168] requires an ECN mark to be equivalent to a drop, both when applied in the network and when responded to by a transport. Unlike Classic ECN marking, i) the network applies L4S marking more immediately and more frequently than drop and ii) the transport response to each mark is reduced and smoothed relative to that for drop. The two changes counterbalance each other so that the throughput of an L4S flow will be roughly the same as a comparable non-L4S flow under the same conditions. Nonetheless, the much more frequent ECN control signals and the finer responses to these signals result in very low queuing delay without compromising link utilization, and this low delay can be maintained during high load. For instance, queuing delay under heavy and highly varying load with the example DCTCP/DualQ solution described below on a DSL or Ethernet link is sub-millisecond on average and roughly 1 to 2 milliseconds at the 99th percentile without losing link utilization [L4Seval22] [DualPI2Linux]. Note that the queuing delay while waiting to acquire a shared medium such as wireless has to be added to the above. It is a different issue that needs to be addressed, but separately (see Section 6.3 of the L4S architecture [RFC9330]).

L4S relies on 'Scalable' congestion controls for these delay properties and for preserving low delay as flow rate scales, hence the name. The congestion control used in Data Center TCP (DCTCP) is an example of a Scalable congestion control, but DCTCP is applicable solely to controlled environments like data centres [RFC8257], because it is too aggressive to coexist with existing TCP-Reno-friendly traffic. Dual-Queue Coupled AQM, which is defined in a complementary Experimental specification [RFC9332], is an AQM framework that enables Scalable congestion controls derived from DCTCP to coexist with existing traffic, each getting roughly the same flow rate when they compete under similar conditions. Note that a Scalable congestion control is still not safe to deploy on the Internet unless it satisfies the requirements listed in Section 4.

L4S is not only for elastic (TCP-like) traffic -- there are Scalable congestion controls for real-time media, such as the L4S variant [SCReAM-L4S] of the SCReAM [RFC8298] RTP Media Congestion Avoidance Techniques (RMCAT). The factor that distinguishes L4S from Classic traffic is its behaviour in response to congestion. The transport wire protocol, e.g., TCP, QUIC, the Stream Control Transmission Protocol (SCTP), the Datagram Congestion Control Protocol (DCCP), or RTP/RTCP, is orthogonal (and therefore not suitable for distinguishing L4S from Classic packets).

The L4S identifier defined in this document is the key piece that distinguishes L4S from 'Classic' (e.g., Reno-friendly) traffic. Then, network bottlenecks can be incrementally modified to distinguish and isolate existing Classic traffic from L4S traffic, to prevent the former from degrading the very low queuing delay and loss of the new Scalable transports, without harming Classic performance at these bottlenecks. Although both sender and network deployment are required before any benefit, initial implementations of the separate parts of the system have been motivated by the potential performance benefits.

1.1. Latency, Loss, and Scaling Problems

Latency is becoming the critical performance factor for many (perhaps most) Internet applications, e.g., interactive web, web services, voice, conversational video, interactive video, interactive remote presence, instant messaging, online gaming, remote desktop, cloud-based applications & services, and remote control of machinery and industrial processes. In many parts of the world, further increases in access network bitrate offer diminishing returns [Dukkipati06], whereas latency is still a multi-faceted problem. As a result, much has been done to reduce propagation time by placing caches or servers closer to users. However, queuing remains a major, albeit intermittent, component of latency.

The Diffserv architecture provides Expedited Forwarding (EF) [RFC3246] so that low-latency traffic can jump the queue of other traffic. If growth in latency-sensitive applications continues, periods with solely latency-sensitive traffic will become increasingly common on links where traffic aggregation is low. During these periods, if all the traffic were marked for the same treatment, Diffserv would make no difference. The links with low aggregation also tend to become the path bottleneck under load, for instance, the access links dedicated to individual sites (homes, small enterprises, or mobile devices). So, instead of differentiation, it becomes imperative to remove the underlying causes of any unnecessary delay.

The Bufferbloat project has shown that excessively large buffering ('bufferbloat') has been introducing significantly more delay than the underlying propagation time [Bufferbloat]. These delays appear only intermittently -- only when a capacity-seeking (e.g., TCP) flow is long enough for the queue to fill the buffer, causing every packet in other flows sharing the buffer to have to work its way through the queue.

AQM was originally developed to solve this problem (and others). Unlike Diffserv, which gives low latency to some traffic at the expense of others, AQM controls latency for _all_ traffic in a class. In general, AQM methods introduce an increasing level of discard from the buffer, the longer the queue persists above a shallow threshold. This gives sufficient signals to capacity-seeking (a.k.a. greedy) flows to keep the buffer empty for its intended purpose: absorbing bursts. However, Random Early Detection (RED) and other algorithms from the 1990s were sensitive to their configuration and hard to set correctly [RFC7567]. So this form of AQM was not widely deployed.

More recent state-of-the-art AQM methods, such as Flow Queue CoDel [RFC8290], Proportional Integral controller Enhanced (PIE) [RFC8033], or Adaptive RED [ARED01], are easier to configure, because they define the queuing threshold in time not bytes, so configuration is invariant whatever the link rate. However, the sawtoothing window of a Classic congestion control creates a dilemma for the operator: either i) configure a shallow AQM operating point so the tips of the sawteeth cause minimal queue delay, but then the troughs underutilize the link, or ii) configure the operating point deeper into the buffer so the troughs utilize the link better, but then the tips cause more delay variation. Even with a perfectly tuned AQM, the additional queuing delay at the tips of the sawteeth will be of the same order as the underlying base round-trip time (RTT), thereby roughly doubling the total RTT.

If a sender's own behaviour is introducing queuing delay variation, no AQM in the network can 'un-vary' the delay without significantly compromising link utilization. Even flow queuing (e.g., [RFC8290]), which isolates one flow from another, cannot isolate a flow from the delay variations it inflicts on itself. Therefore, those applications that need to seek out high bandwidth but also need low latency will have to migrate to Scalable congestion control, which uses much smaller sawtooth variations.

Altering host behaviour is not enough on its own though. Even if hosts adopt low-latency Scalable congestion controls, they need to be isolated from the large queue variations induced by existing Classic congestion controls. L4S AQMs provide that latency isolation in the network, and the L4S identifier enables the AQMs to distinguish the two types of packets that need to be isolated: L4S and Classic. L4S isolation can be achieved with a queue per flow (e.g., [RFC8290]), but a DualQ [RFC9332] is sufficient and actually gives better tail latency [DCttH19]. Both approaches are addressed in this document.

The DualQ solution was developed to make very low latency available without requiring per-flow queues at every bottleneck. This was useful because per-flow queuing (FQ) has well-known downsides -- not least the need to inspect transport-layer headers in the network, which makes it incompatible with privacy approaches such as IPsec Virtual Private Network (VPN) tunnels and incompatible with link-layer queue management, where transport-layer headers can be hidden, e.g., 5G.

Latency is not the only concern addressed by L4S. It was known when TCP congestion avoidance was first developed that it would not scale to high bandwidth-delay products (see footnote 6 of Jacobson and Karels [TCP-CA]). Given that Reno congestion control is already beyond its scaling range at regular broadband bitrates over WAN distances [RFC3649], 'less unscalable' CUBIC [RFC8312] and Compound [CTCP] variants of TCP have been successfully deployed. However, these are now approaching their scaling limits. Unfortunately, fully Scalable congestion controls such as DCTCP [RFC8257] outcompete Classic ECN congestion controls sharing the same queue, which is why they have been confined to private data centres or research testbeds.

It turns out that these Scalable congestion control algorithms that solve the latency problem can also solve the scalability problem of Classic congestion controls. The finer sawteeth in the congestion window (cwnd) have low amplitude, so they cause very little queuing delay variation, and the average time to recover from one congestion signal to the next (the average duration of each sawtooth) remains invariant, which maintains constant tight control as flow rate scales. A background paper [L4Seval22] gives the full explanation of why the design solves both the latency and the scaling problems, both in plain English and in more precise mathematical form. The explanation is summarized without the mathematics in Section 4 of the L4S architecture [RFC9330].

1.2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Classic Congestion Control: A congestion control behaviour that can coexist with standard Reno [RFC5681] without causing significantly negative impact on its flow rate [RFC5033]. With Classic congestion controls, such as Reno or CUBIC, because flow rate has scaled since TCP congestion control was first designed in 1988, it now takes hundreds of round trips (and growing) to recover after a congestion signal (whether a loss or an ECN mark) as shown in the examples in Section 5.1 of the L4S architecture [RFC9330] and in [RFC3649]. Therefore, control of queuing and utilization becomes very slack, and the slightest disturbances (e.g., from new flows starting) prevent a high rate from being attained.

Scalable Congestion Control: A congestion control where the average time from one congestion signal to the next (the recovery time) remains invariant as flow rate scales, all other factors being equal. This maintains the same degree of control over queuing and utilization whatever the flow rate, as well as ensuring that high throughput is robust to disturbances. For instance, DCTCP averages 2 congestion signals per round trip, whatever the flow rate, as do other recently developed Scalable congestion controls, e.g., Relentless TCP [RELENTLESS], Prague for TCP and QUIC [PRAGUE-CC] [PragueLinux], the L4S ECN part of Bottleneck Bandwidth and Round-trip propagation time (BBRv2) [BBRv2] [BBR-CC], and the L4S variant of SCReAM for real-time media [SCReAM-L4S] [RFC8298]. See Section 4.3 for more explanation.

Classic Service: The Classic service is intended for all the congestion control behaviours that coexist with Reno [RFC5681] (e.g., Reno itself, CUBIC [RFC8312], Compound [CTCP], and TFRC [RFC5348]). The term 'Classic queue' means a queue providing the Classic service.

Low Latency, Low Loss, and Scalable throughput (L4S) service: The 'L4S' service is intended for traffic from Scalable congestion control algorithms, such as the Prague congestion control [PRAGUE-CC], which was derived from DCTCP [RFC8257]. The L4S service is for more general traffic than just Prague -- it allows the set of congestion controls with similar scaling properties to Prague to evolve, such as the examples listed above (Relentless, SCReAM, etc.). The term 'L4S queue' means a queue providing the L4S service.

The terms Classic or L4S can also qualify other nouns, such as 'queue', 'codepoint', 'identifier', 'classification', 'packet', and 'flow'. For example, an L4S packet means a packet with an L4S identifier sent from an L4S congestion control.

Both Classic and L4S services can cope with a proportion of unresponsive or less-responsive traffic as well but, in the L4S case, its rate has to be smooth enough or low enough to not build a queue (e.g., DNS, Voice over IP (VoIP), game sync datagrams, etc.).

Reno-friendly: The subset of Classic traffic that is friendly to the standard Reno congestion control defined for TCP in [RFC5681]. The TFRC spec [RFC5348] indirectly implies that 'friendly' is defined as "generally within a factor of two of the sending rate of a TCP flow under the same conditions". Reno-friendly is used here in place of 'TCP-friendly', given the latter has become imprecise, because the TCP protocol is now used with so many different congestion control behaviours, and Reno is used in non-TCP transports, such as QUIC [RFC9000].

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

For L4S, the names used for the four codepoints of the 2-bit IP-ECN field are unchanged from those defined in the ECN spec [RFC3168], i.e., Not-ECT, ECT(0), ECT(1), and CE, where ECT stands for ECN-Capable Transport and CE stands for Congestion Experienced. A packet marked with the CE codepoint is termed 'ECN-marked' or sometimes just 'marked' where the context makes ECN obvious.

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 generalization.

1.3. Scope

The new L4S identifier defined in this specification is applicable for IPv4 and IPv6 packets (as is Classic ECN [RFC3168]). It is applicable for the unicast, multicast, and anycast forwarding modes.

The L4S identifier is an orthogonal packet classification to the Differentiated Services Code Point (DSCP) [RFC2474]. Section 5.4 explains what this means in practice.

This document is Experimental, so it does not update any Standards Track RFCs. Therefore, it depends on [RFC8311], which is a Standards Track specification that:

- updates the ECN Proposed Standard [RFC3168] to allow Experimental RFCs to relax the requirement that an ECN mark must be equivalent to a drop (when the network applies markings and/or when the sender responds to them). For instance, in the Alternative Backoff with ECN (ABE) experiment [RFC8511], this relaxation permits a sender to respond less to ECN marks than to drops;

- changes the status of the Experimental ECN nonce spec [RFC3540] to Historic; and

- makes consequent updates to the following additional Proposed Standard RFCs to reflect the above two bullets:

  - ECN for RTP [RFC6679] and

  - the congestion control specifications of various DCCP Congestion Control Identifier (CCID) profiles [RFC4341] [RFC4342] [RFC5622].

This document is about identifiers that are used for interoperation between hosts and networks. So the audience is broad, covering developers of host transports and network AQMs, as well as covering how operators might wish to combine various identifiers, which would require flexibility from equipment developers.

2. L4S Packet Identification: Document Roadmap

The L4S ECN marking treatment is an experimental alternative to the Classic ECN treatment in [RFC3168], which has been updated by [RFC8311] to allow experiments such as the one defined in the present specification. [RFC4774] discusses some of the issues and evaluation criteria when defining alternative ECN semantics, which are further discussed in Section 4.3.1.

The L4S architecture [RFC9330] describes the three main components of L4S: the sending host behaviour, the marking behaviour in the network, and the L4S ECN protocol that identifies L4S packets as they flow between the two.

Section 3 of this document records the requirements that informed the choice of L4S identifier. Then, subsequent sections specify the L4S ECN protocol, which i) identifies packets that have been sent from hosts that are expected to comply with a broad type of sending behaviour and ii) identifies the marking treatment that network nodes are expected to apply to L4S packets.

For a packet to receive L4S treatment as it is forwarded, the sender sets the ECN field in the IP header to the ECT(1) codepoint. See Section 4 for full transport-layer behaviour requirements, including feedback and congestion response.

A network node that implements the L4S service always classifies arriving ECT(1) packets for L4S treatment and by default classifies CE packets for L4S treatment unless the heuristics described in Section 5.3 are employed. See Section 5 for full network element behaviour requirements, including classification, ECN marking, and interaction of the L4S identifier with other identifiers and per-hop behaviours.

L4S ECN works with ECN tunnelling and encapsulation behaviour as is, except there is one known case where careful attention to configuration is required, which is detailed in Section 6.

This specification of L4S ECN currently has Experimental status. So Section 7 collects the general questions and issues that remain open for investigation during L4S experimentation. Open issues or questions specific to particular components are called out in the specifications of each component part, such as the DualQ [RFC9332].

The IANA assignment of the L4S identifier is specified in Section 8. And Section 9 covers security considerations specific to the L4S identifier. System security aspects, such as policing and privacy, are covered in the L4S architecture [RFC9330].

3. Choice of L4S Packet Identifier: Requirements

This subsection briefly records the process that led to the chosen L4S identifier.

The identifier for packets using the L4S service needs to meet the following requirements:

* it SHOULD survive end to end between source and destination endpoints: across the boundary between host and network, between interconnected networks, and through middleboxes;

* it SHOULD be visible at the IP layer;

* it SHOULD be common to IPv4 and IPv6 and be transport-agnostic;

* it SHOULD be incrementally deployable;

* it SHOULD enable an AQM to classify packets encapsulated by outer IP or lower-layer headers;

* it SHOULD consume minimal extra codepoints; and

* it SHOULD be consistent on all the packets of a transport-layer flow, so that some packets of a flow are not served by a different queue to others.

Whether the identifier would be recoverable if the experiment failed is a factor that could be taken into account. However, this has not been made a requirement, because that would favour schemes that would be easier to fail rather than more likely to succeed.

It is recognized that any choice of identifier is unlikely to satisfy all these requirements, particularly given the limited space left in the IP header. Therefore, a compromise will always be necessary, which is why all the above requirements are expressed with the word "SHOULD" not "MUST".

After extensive assessment of alternative schemes, "ECT(1) and CE codepoints" was chosen as the best compromise. Therefore, this scheme is defined in detail in the following sections, while Appendix B records its pros and cons against the above requirements.

4. Transport-Layer Behaviour (the 'Prague L4S Requirements')

This section defines L4S behaviour at the transport layer, also known as the 'Prague L4S Requirements' (see Appendix A for the origin of the name).

4.1. Codepoint Setting

A sender that wishes a packet to receive L4S treatment as it is forwarded MUST set the ECN field in the IP header (v4 or v6) to the ECT(1) codepoint.

4.2. Prerequisite Transport Feedback

For a transport protocol to provide Scalable congestion control (Section 4.3), it MUST provide feedback of the extent of CE marking on the forward path. When ECN was added to TCP [RFC3168], the feedback method reported no more than one CE mark per round trip. Some transport protocols derived from TCP mimic this behaviour while others report the accurate extent of ECN marking. This means that some transport protocols will need to be updated as a prerequisite for Scalable congestion control. The position for a few well-known transport protocols is given below.

TCP: Support for the accurate ECN feedback requirements [RFC7560] (such as that provided by AccECN [ACCECN]) by both ends is a prerequisite for Scalable congestion control in TCP. Therefore, the presence of ECT(1) in the IP headers even in one direction of a TCP connection will imply that both ends support accurate ECN feedback. However, the converse does not apply. So even if both ends support AccECN, either of the two ends can choose not to use a Scalable congestion control, whatever the other end's choice is.

SCTP: A suitable ECN feedback mechanism for SCTP could add a chunk to report the number of received CE marks (as described in a long-expired document [SCTP-ECN] or as sketched out in Appendix A of the now-obsolete second Standards Track specification of SCTP [RFC4960]).

RTP over UDP: A prerequisite for Scalable congestion control is for both (all) ends of one media-level hop to signal ECN support [RFC6679] and use the new generic RTCP feedback format of [RFC8888]. The presence of ECT(1) implies that both (all) ends of that media-level hop support ECN. However, the converse does not apply. So each end of a media-level hop can independently choose not to use a Scalable congestion control, even if both ends support ECN.

QUIC: Support for sufficiently fine-grained ECN feedback is provided by IETF QUIC transport v1 [RFC9000].

DCCP: The Acknowledgement (ACK) vector in DCCP [RFC4340] is already sufficient to report the extent of CE marking as needed by a Scalable congestion control.

4.3. Prerequisite Congestion Response

As a condition for a host to send packets with the L4S identifier (ECT(1)), it SHOULD implement a congestion control behaviour that ensures that, in steady state, the average duration between induced ECN marks does not increase as flow rate scales up, all other factors being equal. This is termed a Scalable congestion control. This invariant duration ensures that, as flow rate scales, the average period with no feedback information about capacity does not become excessive. It also ensures that queue variations remain small, without having to sacrifice utilization.

With a congestion control that sawtooths to probe capacity, this duration is called the recovery time, because each time the sawtooth yields, on average it takes this time to recover to its previous high point. A Scalable congestion control does not have to sawtooth, but it has to coexist with Scalable congestion controls that do.

For instance, for DCTCP [RFC8257], TCP Prague [PRAGUE-CC] [PragueLinux], and the L4S variant of SCReAM [SCReAM-L4S] [RFC8298], the average recovery time is always half a round trip (or half a reference round trip), whatever the flow rate.

As with all transport behaviours, a detailed specification (probably an Experimental RFC) is expected for each congestion control, following the guidelines for specifying new congestion control algorithms in [RFC5033]. In addition, it is expected that these L4S-specific matters will be documented, specifically the timescale over which the proportionality is averaged and the control of burstiness. The recovery time requirement above is worded as a "SHOULD" rather than a "MUST" to allow reasonable flexibility for such implementations.

The condition 'all other factors being equal' allows the recovery time to be different for different round-trip times, as long as it does not increase with flow rate for any particular RTT.

Saying that the recovery time remains roughly invariant is equivalent to saying that the number of ECN CE marks per round trip remains invariant as flow rate scales, all other factors being equal. For instance, an average recovery time of half of 1 RTT is equivalent to 2 ECN marks per round trip. For those familiar with steady-state congestion response functions, it is also equivalent to say that the congestion window is inversely proportional to the proportion of bytes in packets marked with the CE codepoint (see Section 2 of [PI2]).

In order to coexist safely with other Internet traffic, a Scalable congestion control is not allowed to tag its packets with the ECT(1) codepoint unless it complies with the following numbered requirements and recommendations:

1. A Scalable congestion control MUST be capable of being replaced by a Classic congestion control (by application and/or by administrative control). If a Classic congestion control is activated, it will not tag its packets with the ECT(1) codepoint (see Appendix A.1.3 for rationale).

2. As well as responding to ECN markings, a Scalable congestion control MUST react to packet loss in a way that will coexist safely with Classic congestion controls such as standard Reno [RFC5681], as required by [RFC5033] (see Appendix A.1.4 for rationale).

3. In uncontrolled environments, monitoring MUST be implemented to support detection of problems with an ECN-capable AQM at the path bottleneck that appears not to support L4S and that might be in a shared queue. Such monitoring SHOULD be applied to live traffic that is using Scalable congestion control. Alternatively, monitoring need not be applied to live traffic, if monitoring with test traffic has been arranged to cover the paths that live traffic takes through uncontrolled environments.

   A function to detect the above problems with an ECN-capable AQM MUST also be implemented and used. The detection function SHOULD be capable of making the congestion control adapt its ECN-marking response in real time to coexist safely with Classic congestion controls such as standard Reno [RFC5681], as required by [RFC5033]. This could be complemented by more detailed offline detection of potential problems. If only offline detection is used and potential problems with such an AQM are detected on certain paths, the Scalable congestion control MUST be replaced by a Classic congestion control, at least for the problem paths.

See Section 4.3.1, Appendix A.1.5, and the L4S operational guidance [L4SOPS] for rationale and explanation.

> Note that a Scalable congestion control is not expected to change to setting ECT(0) while it transiently adapts to coexist with Classic congestion controls, whereas a replacement congestion control that solely behaves in the Classic way will set ECT(0).

4. In the range between the minimum likely RTT and typical RTTs expected in the intended deployment scenario, a Scalable congestion control MUST converge towards a rate that is as independent of RTT as is possible without compromising stability or utilization (see Appendix A.1.6 for rationale).

5. A Scalable congestion control SHOULD remain responsive to congestion when typical RTTs over the public Internet are significantly smaller because they are no longer inflated by queuing delay. It would be preferable for the minimum window of a Scalable congestion control to be lower than 1 segment rather than use the timeout approach described for TCP in Section 6.1.2 of the ECN spec [RFC3168] (or an equivalent for other transports). However, a lower minimum is not set as a formal requirement for L4S experiments (see Appendix A.1.7 for rationale).

6. A Scalable congestion control's loss detection SHOULD be resilient to reordering over an adaptive time interval that scales with throughput and adapts to reordering (as in Recent ACK (RACK) [RFC8985]), as opposed to counting only in fixed units of packets (as in the 3 Duplicate ACK (DupACK) rule of NewReno [RFC5681] [RFC6675], which is not scalable). As data rates increase (e.g., due to new and/or improved technology), congestion controls that detect loss by counting in units of packets become more likely to incorrectly treat reordering events as congestion-caused loss events (see Appendix A.1.8 for further rationale). This requirement does not apply to congestion controls that are solely used in controlled environments where the network introduces hardly any reordering.

7. A Scalable congestion control is expected to limit the queue caused by bursts of packets. It would not seem necessary to set the limit any lower than 10% of the minimum RTT expected in a typical deployment (e.g., additional queuing of roughly 250 us for the public Internet). This would be converted to a number of packets by multiplying by the current average packet rate. Then, the queue caused by each burst at the bottleneck link would not exceed 250 us (under the worst-case assumption that the flow is filling the capacity). No normative requirement to limit bursts is given here, and until there is more industry experience from the L4S experiment, it is not even known whether one is needed -- it seems to be in an L4S sender's self-interest to limit bursts.

Each sender in a session can use a Scalable congestion control independently of the congestion control used by the receiver(s) when they send data. Therefore, there might be ECT(1) packets in one direction and ECT(0) or Not-ECT in the other.

Later, this document discusses the conditions for mixing other "'Safe' Unresponsive Traffic" (e.g., DNS, Lightweight Directory Access Protocol (LDAP), NTP, voice, and game sync packets) with L4S traffic; see Section 5.4.1.1. To be clear, although such traffic can share the same queue as L4S traffic, it is not appropriate for the sender to tag it as ECT(1), except in the (unlikely) case that it satisfies the above conditions.

4.3.1. Guidance on Congestion Response in the RFC Series

[RFC3168] requires the congestion responses to a CE-marked packet and a dropped packet to be the same. [RFC8311] is a Standards Track update to [RFC3168] that is intended to enable experimentation with ECN, including the L4S experiment. [RFC8311] allows an experimental congestion control's response to a CE-marked packet to differ from the response to a dropped packet, provided that the differences are documented in an Experimental RFC, such as the present document.

BCP 124 [RFC4774] gives guidance to protocol designers, when specifying alternative semantics for the IP-ECN field. [RFC8311] explained that it did not need to update the best current practice in BCP 124 in order to relax the 'equivalence with drop' requirement because, although BCP 124 quotes the same requirement from [RFC3168], the BCP does not impose requirements based on it. BCP 124 [RFC4774] describes three options for incremental deployment, with Option 3 (in Section 4.3 of BCP 124 [RFC4774]) best matching the L4S case. Option 3's requirement for end-nodes is that they respond to CE marks "in a way that is friendly to flows using IETF-conformant congestion control." This echoes other general congestion control requirements in the RFC Series, for example, [RFC5033] states that "...congestion controllers that have a significantly negative impact on traffic using standard congestion control may be suspect" and [RFC8085], which concerns UDP congestion control, states that "Bulk-transfer applications that choose not to implement TFRC or TCP-like windowing SHOULD implement a congestion control scheme that results in bandwidth (capacity) use that competes fairly with TCP within an order of magnitude."

The normative Item 3 in Section 4.3 above (which concerns L4S response to congestion from a Classic ECN AQM) aims to ensure that these 'coexistence' requirements are satisfied, but it makes some compromises. This subsection highlights and justifies those compromises, and Appendix A.1.5 and the L4S operational guidance [L4SOPS] give detailed analysis, examples, and references (the normative text in that bullet takes precedence if any informative elaboration leads to ambiguity). The approach is based on an assessment of the risk of harm, which is a combination of the prevalence of the conditions necessary for harm to occur, and the potential severity of the harm if they do.

Prevalence: There are three cases:

* Drop Tail: Coexistence between L4S and Classic flows is not in doubt where the bottleneck does not support any form of ECN,

which has remained by far the most prevalent case since the ECN spec [RFC3168] was published in 2001.

- L4S: Coexistence is not in doubt if the bottleneck supports L4S.

- Classic ECN: The compromises centre around cases where the bottleneck supports Classic ECN [RFC3168] but not L4S. But it depends on which sub-case:

  - Shared Queue with Classic ECN: At the time of writing, the members of the Transport Working Group are not aware of any current deployments of single-queue Classic ECN bottlenecks in the Internet. Nonetheless, at the scale of the Internet, rarity need not imply small numbers nor that there will be rarity in the future.

  - Per-Flow Queues with Classic ECN: Most AQMs with per-flow queuing deployed from 2012 onwards had Classic ECN enabled by default, specifically FQ-CoDel [RFC8290] and COBALT [COBALT]. But the compromises only apply to the second of two further sub-cases:

    - With per-flow queuing, coexistence between Classic and L4S flows is not normally a problem, because different flows are not meant to be in the same queue (BCP 124 [RFC4774] did not foresee the introduction of per-flow queuing, which appeared as a potential isolation technique some eight years after the BCP was published).

    - However, the isolation between L4S and Classic flows is not perfect in cases where the hashes of flow identifiers (IDs) collide or where multiple flows within a Layer 3 VPN are encapsulated within one flow ID.

To summarize, the coexistence problem is confined to cases of imperfect flow isolation in an FQ or in potential cases where a Classic ECN AQM has been deployed in a shared queue (see the L4S operational guidance [L4SOPS] for further details including recent surveys attempting to quantify prevalence). Further, if one of these cases does occur, the coexistence problem does not arise unless sources of Classic and L4S flows are simultaneously sharing the same bottleneck queue (e.g., different applications in the same household), and flows of each type have to be large enough to coincide for long enough for any throughput imbalance to have developed. Therefore, how often the coexistence problem arises in practice is listed in Section 7 as an open question that L4S experiments will need to answer.

Severity: Where long-running L4S and Classic flows coincide in a shared queue, testing of one L4S congestion control (TCP Prague) has found that the imbalance in average throughput between an L4S and a Classic flow can reach 25:1 in favour of L4S in the worst case [ecn-fallback]. However, when capacity is most scarce, the Classic flow gets a higher proportion of the link, for instance, over a 4 Mb/s link the throughput ratio is below ~10:1 over paths with a base RTT below 100 ms, and it falls below ~5:1 for base RTTs below 20 ms.

These throughput ratios can clearly fall well outside current RFC guidance on coexistence. However, the tendency towards leaving a greater share for Classic flows at lower link rate and the very limited prevalence of the conditions necessary for harm to occur led to the possibility of allowing the RFC requirements to be compromised, albeit briefly:

* The recommended approach is still to detect and adapt to a Classic ECN AQM in real time, which is fully consistent with all the RFCs on coexistence. In other words, the "SHOULD"s in Item 3 of Section 4.3 above expect the sender to implement something similar to the proof-of-concept code that detects the presence of a Classic ECN AQM and falls back to a Classic congestion response within a few round trips [ecn-fallback]. However, although this code reliably detects a Classic ECN AQM, the current code can also wrongly categorize an L4S AQM as Classic, most often in cases when link rate is low or RTT is high. Although this is the safe way round, and although implementers are expected to be able to improve on this proof of concept, concerns have been raised that implementers might lose faith in such detection and disable it.

* Item 3 in Section 4.3 above therefore allows a compromise where coexistence could briefly diverge from the requirements in the RFC Series, but mandatory monitoring is required in order to detect such cases and trigger remedial action. This approach tolerates a brief divergence from the RFCs given the likely low prevalence and given harm here means a flow progresses more slowly than it would otherwise, but it does progress. The L4S operational guidance [L4SOPS] outlines a range of example remedial actions that include alterations to either the sender or the network. However, the final normative requirement in Item 3 of Section 4.3 above places ultimate responsibility for remedial action on the sender. If coexistence problems with a Classic ECN AQM are detected (implying they have not been resolved by the network), it states that the sender "MUST" revert to a Classic congestion control.

[L4SOPS] also gives example ways in which L4S congestion controls can be rolled out initially in lower-risk scenarios.

4.4. Filtering or Smoothing of ECN Feedback

Section 5.2 below specifies that an L4S AQM is expected to signal L4S ECN immediately, to avoid introducing delay due to filtering or smoothing. This contrasts with a Classic AQM, which filters out variations in the queue before signalling ECN marking or drop. In the L4S architecture [RFC9330], responsibility for smoothing out these variations shifts to the sender's congestion control.

This shift of responsibility has the advantage that each sender can smooth variations over a timescale proportionate to its own RTT. Whereas, in the Classic approach, the network doesn't know the RTTs of any of the flows, so it has to smooth out variations for a worst-case RTT to ensure stability. For all the typical flows with shorter

RTTs than the worst-case, this makes congestion control unnecessarily sluggish.

This also gives an L4S sender the choice not to smooth, depending on its context (start-up, congestion avoidance, etc.). Therefore, this document places no requirement on an L4S congestion control to smooth out variations in any particular way. Implementers are encouraged to openly publish the approach they take to smoothing as well as results and experience they gain during the L4S experiment.

5. Network Node Behaviour

5.1. Classification and Re-Marking Behaviour

A network node that implements the L4S service:

* MUST classify arriving ECT(1) packets for L4S treatment, unless overridden by another classifier (e.g., see Section 5.4.1.2).

* MUST classify arriving CE packets for L4S treatment as well, unless overridden by another classifier or unless the exception referred to next applies.

  CE packets might have originated as ECT(1) or ECT(0), but the above rule to classify them as if they originated as ECT(1) is the safe choice (see Appendix B for rationale). The exception is where some flow-aware in-network mechanism happens to be available for distinguishing CE packets that originated as ECT(0), as described in Section 5.3, but there is no implication that such a mechanism is necessary.

An L4S AQM treatment follows similar codepoint transition rules to those in [RFC3168]. Specifically, the ECT(1) codepoint MUST NOT be changed to any codepoint other than CE, and CE MUST NOT be changed to any other codepoint. An ECT(1) packet is classified as 'ECN-capable', and if congestion increases, an L4S AQM algorithm will increasingly mark the IP-ECN field as CE, otherwise forwarding packets unchanged as ECT(1). Necessary conditions for an L4S marking treatment are defined in Section 5.2.

Under persistent overload, an L4S marking treatment MUST begin applying drop to L4S traffic until the overload episode has subsided, as recommended for all AQM methods in Section 4.2.1 of [RFC7567], which follows the similar advice in Section 7 of [RFC3168]. During overload, it MUST apply the same drop probability to L4S traffic as it would to Classic traffic.

Where an L4S AQM is transport-aware, this requirement could be satisfied by using drop in only the most overloaded individual per-flow AQMs. In a DualQ with flow-aware queue protection (e.g., [DOCSIS-QPROT]), this could be achieved by redirecting packets in those flows contributing most to the overload out of the L4S queue so that they are subjected to drop in the Classic queue.

For backward compatibility in uncontrolled environments, a network node that implements the L4S treatment MUST also implement an AQM treatment for the Classic service as defined in Section 1.2. This Classic AQM treatment need not mark ECT(0) packets, but if it does, see Section 5.2 for the strengths of the markings relative to drop. It MUST classify arriving ECT(0) and Not-ECT packets for treatment by this Classic AQM (for the DualQ Coupled AQM; see the extensive discussion on classification in Sections 2.3 and 2.5.1.1 of [RFC9332]).

In case unforeseen problems arise with the L4S experiment, it MUST be possible to configure an L4S implementation to disable the L4S treatment. Once disabled, ECT(1) packets MUST be treated as if they were Not-ECT.

5.2. The Strength of L4S CE Marking Relative to Drop

The relative strengths of L4S CE and drop are irrelevant where AQMs are implemented in separate queues per application-flow, which are then explicitly scheduled (e.g., with an FQ scheduler as in FQ-CoDel [RFC8290]). Nonetheless, the relationship between them needs to be defined for the coupling between L4S and Classic congestion signals in a DualQ Coupled AQM [RFC9332], as indicated below.

Unless an AQM node schedules application flows explicitly, the likelihood that the AQM drops a Not-ECT Classic packet (p_C) MUST be roughly proportional to the square of the likelihood that it would have marked it if it had been an L4S packet (p_L). That is:

```
p_C ~= (p_L / k)^2
```

The constant of proportionality (k) does not have to be standardized for interoperability, but a value of 2 is RECOMMENDED. The term 'likelihood' is used above to allow for marking and dropping to be either probabilistic or deterministic.

This formula ensures that Scalable and Classic flows will converge to roughly equal congestion windows, for the worst case of Reno congestion control. This is because the congestion windows of Scalable and Classic congestion controls are inversely proportional to p_L and sqrt(p_C), respectively. So squaring p_C in the above formula counterbalances the square root that characterizes Reno-friendly flows.

> Note that, contrary to [RFC3168], an AQM implementing the L4S and Classic treatments does not mark an ECT(1) packet under the same conditions that it would have dropped a Not-ECT packet, as allowed by [RFC8311], which updates [RFC3168]. However, if it marks ECT(0) packets, it does so under the same conditions that it would have dropped a Not-ECT packet [RFC3168].

Also, in the L4S architecture [RFC9330], the sender, not the network, is responsible for smoothing out variations in the queue. So an L4S AQM MUST signal congestion as soon as possible. Then, an L4S sender generally interprets CE marking as an unsmoothed signal.

This requirement does not prevent an L4S AQM from mixing in additional congestion signals that are smoothed, such as the signals from a Classic smoothed AQM that are coupled with unsmoothed L4S signals in the coupled DualQ [RFC9332], but only as long as the onset of congestion can be signalled immediately and can be interpreted by the sender as if it has been signalled immediately, which is important for interoperability

5.3. Exception for L4S Packet Identification by Network Nodes with Transport-Layer Awareness

To implement L4S packet classification, a network node does not need to identify transport-layer flows. Nonetheless, if an L4S network node classifies packets by their transport-layer flow ID and their ECN field, and if all the ECT packets in a flow have been ECT(0), the node MAY classify any CE packets in the same flow as if they were Classic ECT(0) packets. In all other cases, a network node MUST classify all CE packets as if they were ECT(1) packets. Examples of such other cases are: i) if no ECT packets have yet been identified in a flow; ii) if it is not desirable for a network node to identify transport-layer flows; or iii) if some ECT packets in a flow have been ECT(1) (this advice will need to be verified as part of L4S experiments).

5.4. Interaction of the L4S Identifier with Other Identifiers

The examples in this section concern how additional identifiers might complement the L4S identifier to classify packets between class-based queues. Firstly, Section 5.4.1 considers two queues, L4S and Classic, as in the DualQ Coupled AQM [RFC9332], either alone (Section 5.4.1.1) or within a larger queuing hierarchy (Section 5.4.1.2). Then, Section 5.4.2 considers schemes that might combine per-flow 5-tuples with other identifiers.

5.4.1. DualQ Examples of Other Identifiers Complementing L4S Identifiers

5.4.1.1. Inclusion of Additional Traffic with L4S

In a typical case for the public Internet, a network element that implements L4S in a shared queue might want to classify some low-rate but unresponsive traffic (e.g., DNS, LDAP, NTP, voice, and game sync packets) into the low-latency queue to mix with L4S traffic. In this case, it would not be appropriate to call the queue an L4S queue, because it is shared by L4S and non-L4S traffic. Instead, it will be called the low-latency or L queue. The L queue then offers two different treatments:

* the L4S treatment, which is a combination of the L4S AQM treatment and a priority scheduling treatment, and

* the low-latency treatment, which is solely the priority scheduling treatment, without ECN marking by the AQM.

To identify packets for just the scheduling treatment, it would be inappropriate to use the L4S ECT(1) identifier, because such traffic is unresponsive to ECN marking. Examples of relevant non-ECN identifiers are:

- address ranges of specific applications or hosts configured to be, or known to be, safe, e.g., hard-coded Internet of Things (IoT) devices sending low-intensity traffic;

- certain low data-volume applications or protocols (e.g., ARP and DNS); and

- specific Diffserv codepoints that indicate traffic with limited burstiness such as the EF [RFC3246], VOICE-ADMIT [RFC5865], or proposed Non-Queue-Building (NQB) [NQB-PHB] service classes or equivalent Local-use DSCPs (see [L4S-DIFFSERV]).

To be clear, classifying into the L queue based on application-layer identification (e.g., DNS) is an example of a local optimization, not a recommendation. Applications will not be able to rely on such unsolicited optimization. A more reliable approach would be for the sender to set an appropriate IP-layer identifier, such as one of the above Diffserv codepoints.

In summary, a network element that implements L4S in a shared queue MAY classify additional types of packets into the L queue based on identifiers other than the IP-ECN field, but the types SHOULD be 'safe' to mix with L4S traffic, where 'safe' is explained in Section 5.4.1.1.1.

A packet that carries one of these non-ECN identifiers to classify it into the L queue would not be subject to the L4S ECN-marking treatment, unless it also carried an ECT(1) or CE codepoint. The specification of an L4S AQM MUST define the behaviour for packets with unexpected combinations of codepoints, e.g., a non-ECN-based classifier for the L queue but with ECT(0) in the IP-ECN field (for examples with appropriate behaviours, see Section 2.5.1.1 of the DualQ spec [RFC9332]).

For clarity, non-ECN identifiers, such as the examples itemized above, might be used by some network operators who believe they identify non-L4S traffic that would be safe to mix with L4S traffic. They are not alternative ways for a host to indicate that it is sending L4S packets. Only the ECT(1) ECN codepoint indicates to a network element that a host is sending L4S packets (and CE indicates that it could have originated as ECT(1)). Specifically, ECT(1) indicates that the host claims its behaviour satisfies the prerequisite transport requirements in Section 4.

In order to include non-L4S packets in the L queue, a network node MUST NOT change Not-ECT or ECT(0) in the IP-ECN field into an L4S identifier. This ensures that these codepoints survive for any potential use later on the network path. If a non-compliant or malicious network node did swap ECT(0) to ECT(1), the packet could subsequently be ECN-marked by a downstream L4S AQM, but the sender would respond to congestion indications thinking it had sent a Classic packet. This could result in the flow yielding excessively to other L4S flows sharing the downstream bottleneck.

5.4.1.1.1. 'Safe' Unresponsive Traffic

The above section requires unresponsive traffic to be 'safe' to mix with L4S traffic. Ideally, this means that the sender never sends any sequence of packets at a rate that exceeds the available capacity of the bottleneck link. However, typically an unresponsive transport does not even know the bottleneck capacity of the path, let alone its available capacity. Nonetheless, an application can be considered safe enough if it paces packets out (not necessarily with absolute regularity) such that its maximum instantaneous rate from packet to packet stays well below a typical broadband access rate.

This is a vague but useful definition, because many low-latency applications of interest, such as DNS, voice, game sync packets, RPC, ACKs, and keep-alives, could match this description.

Low-rate streams, such as voice and game sync packets, might not use continuously adapting ECN-based congestion control, but they ought to at least use a 'circuit-breaker' style of congestion response [RFC8083]. If the volume of traffic from unresponsive applications is high enough to overload the link, this will at least protect the capacity available to responsive applications. However, queuing delay in the L queue would probably then rise to the typically higher level targeted by a Classic (drop-based) AQM. If a network operator considers that such self-restraint is not enough, it might want to police the L queue (see Section 8.2 of the L4S architecture [RFC9330]).

5.4.1.2. Exclusion of Traffic from L4S Treatment

To extend the above example, an operator might want to exclude some traffic from the L4S treatment for a policy reason, e.g., security (traffic from malicious sources) or commercial (e.g., the operator may wish to initially confine the benefits of L4S to business customers).

In this exclusion case, the classifier MUST classify on the relevant locally used identifiers (e.g., source addresses) before classifying the non-matching traffic on the end-to-end L4S ECN identifier.

A network node MUST NOT alter the end-to-end L4S ECN identifier from L4S to Classic, because an operator decision to exclude certain traffic from L4S treatment is local-only. The end-to-end L4S identifier then survives for other operators to use, or indeed, they can apply their own policy, independently based on their own choice of locally used identifiers. This approach also allows any operator to remove its locally applied exclusions in future, e.g., if it wishes to widen the benefit of the L4S treatment to all its customers. If a non-compliant or malicious network node did swap ECT(1) to ECT(0), the packet could subsequently be ECN-marked by a downstream Classic ECN AQM. L4S senders are required to detect and handle such treatment (see Item 3 in Section 4.3), but that does not make this swap OK, because such detection is not known to be perfect or immediate.

A network node that supports L4S but excludes certain packets carrying the L4S identifier from L4S treatment MUST still apply marking or dropping that is compatible with an L4S congestion response. For instance, it could either drop such packets with the same likelihood as Classic packets or ECN-mark them with a likelihood appropriate to L4S traffic (e.g., the coupled probability in a DualQ Coupled AQM) but aiming for the Classic delay target. It MUST NOT ECN-mark such packets with a Classic marking probability, which could confuse the sender.

5.4.1.3. Generalized Combination of L4S and Other Identifiers

L4S concerns low latency, which it can provide for all traffic without differentiation and without _necessarily_ affecting bandwidth allocation. Diffserv provides for differentiation of both bandwidth and low latency, but its control of latency depends on its control of bandwidth. L4S and Diffserv can be combined if a network operator wants to control bandwidth allocation but also wants to provide low latency, i.e., for any amount of traffic within one of these allocations of bandwidth (rather than only providing low latency by limiting bandwidth) [L4S-DIFFSERV].

The DualQ examples so far have been framed in the context of providing the default Best Effort Per-Hop Behaviour (PHB) using two queues -- a low-latency (L) queue and a Classic (C) queue. This single DualQ structure is expected to be the most common and useful arrangement. But, more generally, an operator might choose to control bandwidth allocation through a hierarchy of Diffserv PHBs at a node and to offer one (or more) of these PHBs using a pair of queues for a low latency and a Classic variant of the PHB.

In the first case, if we assume that a network element provides no PHBs except the DualQ, if a packet carries ECT(1) or CE, the network element would classify it for the L4S treatment irrespective of its DSCP. And, if a packet carried (for example) the EF DSCP, the network element could classify it into the L queue irrespective of its ECN codepoint. However, where the DualQ is in a hierarchy of other PHBs, the classifier would classify some traffic into other PHBs based on DSCP before classifying between the low-latency and Classic queues (based on ECT(1), CE, and perhaps also the EF DSCP or other identifiers as in the above example). [L4S-DIFFSERV] gives a number of examples of such arrangements to address various requirements.

[L4S-DIFFSERV] describes how an operator might use L4S to offer low latency as well as Diffserv for bandwidth differentiation. It identifies two main types of approach, which can be combined: the operator might split certain Diffserv PHBs between L4S and a corresponding Classic service. Or it might split the L4S and/or the Classic service into multiple Diffserv PHBs. In either of these cases, a packet would have to be classified on its Diffserv and ECN codepoints.

In summary, there are numerous ways in which the L4S ECN identifier (ECT(1) and CE) could be combined with other identifiers to achieve particular objectives. The following categorization articulates those that are valid, but it is not necessarily exhaustive. Those tagged as 'Recommended-standard-use' could be set by the sending host or a network. Those tagged as 'Local-use' would only be set by a network:

1. Identifiers Complementing the L4S Identifier

   a. Including More Traffic in the L Queue

      (could use Recommended-standard-use or Local-use identifiers)

   b. Excluding Certain Traffic from the L Queue

      (Local-use only)

2. Identifiers to Place L4S Classification in a PHB Hierarchy

   (could use Recommended-standard-use or Local-use identifiers)

   a. PHBs before L4S ECN Classification

   b. PHBs after L4S ECN Classification

5.4.2. Per-flow Queuing Examples of Other Identifiers Complementing L4S Identifiers

At a node with per-flow queuing (e.g., FQ-CoDel [RFC8290]), the L4S identifier could complement the transport-layer flow ID as a further level of flow granularity (i.e., Not-ECT and ECT(0) queued separately from ECT(1) and CE packets). In Appendix B, the "Risk of reordering Classic CE packets within a flow" discusses the resulting ambiguity if packets originally set to ECT(0) are marked CE by an upstream AQM before they arrive at a node that classifies CE as L4S. It argues that the risk of reordering is vanishingly small, and the consequence of such a low level of reordering is minimal.

Alternatively, it could be assumed that it is not in a flow's own interest to mix Classic and L4S identifiers. Then, the AQM could use the IP-ECN field to switch itself between a Classic and an L4S AQM behaviour within one per-flow queue. For instance, for ECN-capable packets, the AQM might consist of a simple marking threshold, and an L4S ECN identifier might simply select a shallower threshold than a Classic ECN identifier would.

5.5. Limiting Packet Bursts from Links

As well as senders needing to limit packet bursts (Section 4.3), links need to limit the degree of burstiness they introduce. In both cases (senders and links), this is a trade-off, because batch-handling of packets is done for good reason, e.g., for processing efficiency or to make efficient use of medium acquisition delay. Some take the attitude that there is no point reducing burst delay at the sender below that introduced by links (or vice versa). However, delay reduction proceeds by cutting down 'the longest pole in the tent', which turns the spotlight on the next longest, and so on.

This document does not set any quantified requirements for links to limit burst delay, primarily because link technologies are outside the remit of L4S specifications. Nonetheless, the following two subsections outline opportunities for addressing bursty links in the process of L4S implementation and deployment.

5.5.1. Limiting Packet Bursts from Links Fed by an L4S AQM

It would not make sense to implement an L4S AQM that feeds into a particular link technology without also reviewing opportunities to reduce any form of burst delay introduced by that link technology. This would at least limit the bursts that the link would otherwise introduce into the onward traffic, which would cause jumpy feedback to the sender as well as potential extra queuing delay downstream. This document does not presume to even give guidance on an appropriate target for such burst delay until there is more industry experience of L4S. However, as suggested in Section 4.3, it would not seem necessary to limit bursts lower than roughly 10% of the minimum base RTT expected in the typical deployment scenario (e.g., 250 us burst duration for links within the public Internet).

5.5.2. Limiting Packet Bursts from Links Upstream of an L4S AQM

The initial scope of the L4S experiment is to deploy L4S AQMs at bottlenecks and L4S congestion controls at senders. This is expected to highlight interactions with the most bursty upstream links and lead operators to tune down the burstiness of those links in their networks that are configurable or, failing that, to have to compromise on the delay target of some L4S AQMs. It might also require specific redesign work relevant to the most problematic link types. Such knock-on effects of initial L4S deployment would all be a part of the learning from the L4S experiment.

The details of such link changes are beyond the scope of the present document. Nonetheless, where L4S technology is being implemented on an outgoing interface of a device, it would make sense to consider opportunities for reducing bursts arriving at other incoming interfaces. For instance, where an L4S AQM is implemented to feed into the upstream WAN interface of a home gateway, there would be opportunities to alter the Wi-Fi profiles sent out of any Wi-Fi interfaces from the same device, in order to mitigate incoming bursts of aggregated Wi-Fi frames from other Wi-Fi stations.

6. Behaviour of Tunnels and Encapsulations

6.1. No Change to ECN Tunnels and Encapsulations in General

The L4S identifier is expected to work through and within any tunnel without modification, as long as the tunnel propagates the ECN field in any of the ways that have been defined since the first variant in the year 2001 [RFC3168]. L4S will also work with (but does not rely on) any of the more recent updates to ECN propagation in [RFC4301], [RFC6040], or [ECN-SHIM]. However, it is likely that some tunnels still do not implement ECN propagation at all. In these cases, L4S will work through such tunnels, but within them the outer header of L4S traffic will appear as Classic.

AQMs are typically implemented where an IP-layer buffer feeds into a lower layer, so they are agnostic to link-layer encapsulations. Where a bottleneck link is not IP-aware, the L4S identifier is still expected to work within any lower-layer encapsulation without modification, as long it propagates the IP-ECN field as defined for the link technology, for example, for MPLS [RFC5129] or Transparent Interconnection of Lots of Links (TRILL) [TRILL-ECN-SUPPORT]. In some of these cases, e.g., Layer 3 Ethernet switches, the AQM accesses the IP-layer header within the outer encapsulation, so again the L4S identifier is expected to work without modification. Nonetheless, the programme to define ECN for other lower layers is still in progress [ECN-ENCAP].

6.2. VPN Behaviour to Avoid Limitations of Anti-Replay

If a mix of L4S and Classic packets is sent into the same security association (SA) of a VPN, and if the VPN egress is employing the optional anti-replay feature, it could inappropriately discard Classic packets (or discard the records in Classic packets) by mistaking their greater queuing delay for a replay attack (see "Dropped Packets for Tunnels with Replay Protection Enabled" in [Heist21] for the potential performance impact). This known problem is common to both IPsec [RFC4301] and DTLS [RFC9147] VPNs, given they use similar anti-replay window mechanisms. The mechanism used can only check for replay within its window, so if the window is smaller than the degree of reordering, it can only assume there might be a replay attack and discard all the packets behind the trailing edge of the window. The specifications of IPsec Authentication Header (AH) [RFC4302] and Encapsulating Security Payload (ESP) [RFC4303] suggest that an implementer scales the size of the anti-replay window with interface speed, and DTLS v1.3 [RFC9147] states that "The receiver SHOULD pick a window large enough to handle any plausible reordering, which depends on the data rate." However, in practice, the size of a VPN's anti-replay window is not always scaled appropriately.

If a VPN carrying traffic participating in the L4S experiment experiences inappropriate replay detection, the foremost remedy would be to ensure that the egress is configured to comply with the above window-sizing requirements.

If an implementation of a VPN egress does not support a sufficiently large anti-replay window, e.g., due to hardware limitations, one of the temporary alternatives listed in order of preference below might be feasible instead:

* If the VPN can be configured to classify packets into different SAs indexed by DSCP, apply the appropriate locally defined DSCPs to Classic and L4S packets. The DSCPs could be applied by the network (based on the least-significant bit of the IP-ECN field), or by the sending host. Such DSCPs would only need to survive as far as the VPN ingress.

* If the above is not possible and it is necessary to use L4S, either of the following might be appropriate as a last resort:

  - disable anti-replay protection at the VPN egress, after considering the security implications (it is mandatory to allow the anti-replay facility to be disabled in both IPsec and DTLS), or

  - configure the tunnel ingress not to propagate ECN to the outer, which would lose the benefits of L4S and Classic ECN over the VPN.

Modification to VPN implementations is outside the present scope, which is why this section has so far focused on reconfiguration. Although this document does not define any requirements for VPN implementations, determining whether there is a need for such requirements could be one aspect of L4S experimentation.

7. L4S Experiments

This section describes open questions that L4S experiments ought to focus on. This section also documents outstanding open issues that will need to be investigated as part of L4S experimentation, given they could not be fully resolved during the working group phase. It also lists metrics that will need to be monitored during experiments (summarizing text elsewhere in L4S documents) and finally lists some potential future directions that researchers might wish to investigate.

In addition to this section, i) the DualQ spec [RFC9332] sets operational and management requirements for experiments with DualQ Coupled AQMs, and ii) general operational and management requirements for experiments with L4S congestion controls are given in Sections 4 and 5 above, e.g., coexistence and scaling requirements and incremental deployment arrangements.

The specification of each Scalable congestion control will need to include protocol-specific requirements for configuration and monitoring performance during experiments. Appendix A of [RFC5706] provides a helpful checklist.

7.1. Open Questions

L4S experiments would be expected to answer the following questions:

* Have all the parts of L4S been deployed, and if so, what proportion of paths support it?

  - What types of L4S AQMs were deployed, e.g., FQ, coupled DualQ, uncoupled DualQ, other? And how prevalent was each?

  - Are the signalling patterns emitted by the deployed AQMs in any way different from those expected when the Prague requirements for endpoints were written?

* Does use of L4S over the Internet result in a significantly improved user experience?

* Has L4S enabled novel interactive applications?

* Did use of L4S over the Internet result in improvements to the

following metrics:

  - queue delay (mean and 99th percentile) under various loads;
  - utilization;
  - starvation / fairness; and
  - scaling range of flow rates and RTTs?

* How dependent was the performance of L4S service on the bottleneck bandwidth or the path RTT?

* How much do bursty links in the Internet affect L4S performance (see "Underutilization with Bursty Links" in [Heist21]) and how prevalent are they? How much limitation of burstiness from upstream links was needed and/or was realized -- both at senders and at links, especially radio links -- or how much did L4S target delay have to be increased to accommodate the bursts (see Item 7 in Section 4.3 and see Section 5.5.2)?

* Is the initial experiment with mis-identified bursty traffic at high RTT (see "Underutilization with Bursty Traffic" in [Heist21]) indicative of similar problems at lower RTTs, and if so, how effective is the suggested remedy in Appendix A.1 of the DualQ spec [RFC9332] (or possible other remedies)?

* Was per-flow queue protection typically (un)necessary?

  - How well did overload protection or queue protection work?

* How well did L4S flows coexist with Classic flows when sharing a bottleneck?

  - How frequently did problems arise?
  - What caused any coexistence problems, and were any problems due to single-queue Classic ECN AQMs (this assumes single-queue Classic ECN AQMs can be distinguished from FQ ones)?

* How prevalent were problems with the L4S service due to tunnels/ encapsulations that do not support ECN decapsulation?

* How easy was it to implement a fully compliant L4S congestion control, over various different transport protocols (TCP, QUIC, RMCAT, etc.)?

Monitoring for harm to other traffic, specifically bandwidth starvation or excess queuing delay, will need to be conducted alongside all early L4S experiments. It is hard, if not impossible, for an individual flow to measure its impact on other traffic. So such monitoring will need to be conducted using bespoke monitoring across flows and/or across classes of traffic.

7.2. Open Issues

- What is the best way forward to deal with L4S over single-queue Classic ECN AQM bottlenecks, given current problems with misdetecting L4S AQMs as Classic ECN AQMs? See the L4S operational guidance [L4SOPS].

- Fixing the poor interaction between current L4S congestion controls and CoDel with only Classic ECN support during flow startup. Originally, this was due to a bug in the initialization of the congestion average in the Linux implementation of TCP Prague. That was quickly fixed, which removed the main performance impact, but further improvement would be useful (by modifying either CoDel or Scalable congestion controls, or both).

7.3. Future Potential

Researchers might find that L4S opens up the following interesting areas for investigation:

- potential for faster convergence time and tracking of available capacity;

- potential for improvements to particular link technologies and cross-layer interactions with them;

- potential for using virtual queues, e.g., to further reduce latency jitter or to leave headroom for capacity variation in radio networks;

- development and specification of reverse path congestion control using L4S building blocks (e.g., AccECN or QUIC);

- once queuing delay is cut down, what becomes the 'second-longest pole in the tent' (other than the speed of light)?

- novel alternatives to the existing set of L4S AQMs; and

- novel applications enabled by L4S.

8. IANA Considerations

The semantics of the 01 codepoint of the ECN field of the IP header are specified by this Experimental RFC. The process for an Experimental RFC to assign this codepoint in the IP header (v4 and v6) is documented in Proposed Standard [RFC8311], which updates the Proposed Standard [RFC3168].

IANA has updated the 01 entry in the "ECN Field (Bits 6-7)" registry (see <https://www.iana.org/assignments/dscp-registry/>) as follows:

Binary	Keyword	Reference
01	ECT(1) (ECN-Capable Transport(1))[1]	[RFC8311] [RFC Errata 5399] RFC 9331

Table 1: ECN Field (Bits 6-7) Registry

[1] ECT(1) is for experimental use only [RFC8311], Section 4.2

9. Security Considerations

Approaches to assure the integrity of signals using the new identifier are introduced in Appendix C.1. See the security considerations in the L4S architecture [RFC9330] for further discussion of misuse of the identifier, as well as extensive discussion of policing rate and latency in regard to L4S.

Defining ECT(1) as the L4S identifier introduces a difference between the effects of ECT(0) and ECT(1) that [RFC3168] previously defined as distinct but with equivalent effect. For L4S ECN, a network node is still required not to swap one to the other, even if the network operator chooses to locally apply the treatment associated with the opposite codepoint (see Sections 5.4.1.1 and 5.4.1.2). These sections also describe the potential effects if a non-compliant or malicious network node does swap one to the other. The present specification does not change the effects of other unexpected transitions of the IP-ECN field, which are still as described in Section 18 of [RFC3168].

If the anti-replay window of a VPN egress is too small, it will mistake deliberate delay differences as a replay attack and discard higher-delay packets (e.g., Classic) carried within the same security association (SA) as low-delay packets (e.g., L4S). Section 6.2 recommends that VPNs used in L4S experiments are configured with a sufficiently large anti-replay window, as required by the relevant specifications. It also discusses other alternatives.

If a user taking part in the L4S experiment sets up a VPN without being aware of the above advice, and if the user allows anyone to send traffic into their VPN, they would open up a DoS vulnerability in which an attacker could induce the VPN's anti-replay mechanism to discard enough of the user's Classic (C) traffic (if they are receiving any) to cause a significant rate reduction. While the user is actively downloading C traffic, the attacker sends C traffic into the VPN to fill the remainder of the bottleneck link, then sends intermittent L4S packets to maximize the chance of exceeding the VPN's replay window. The user can prevent this attack by following the recommendations in Section 6.2.

The recommendation to detect loss in time units prevents the ACK-splitting attacks described in [Savage-TCP].

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>.

[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>.

[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>.

[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>.

10.2. Informative References

[A2DTCP] Zhang, T., Wang, J., Huang, J., Huang, Y., Chen, J., and Y. Pan, "Adaptive-Acceleration Data Center TCP", IEEE Transactions on Computers, Volume 64, Issue 6, pp. 1522-1533, DOI 10.1109/TC.2014.2345393, June 2015, <https://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6871352>.

[ACCECN] Briscoe, B., Kühlewind, M., and R. Scheffenegger, "More Accurate ECN Feedback in TCP", Work in Progress, Internet-Draft, draft-ietf-tcpm-accurate-ecn-22, 9 November 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-tcpm-accurate-ecn-22>.

[Ahmed19] Ahmed, A.S., "Extending TCP for Low Round Trip Delay", Master's Thesis, University of Oslo, August 2019, <https://www.duo.uio.no/handle/10852/70966>.

[Alizadeh-stability]
Alizadeh, M., Javanmard, A., and B. Prabhakar, "Analysis of DCTCP: Stability, Convergence, and Fairness", SIGMETRICS '11: Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems, pp. 73-84, DOI 10.1145/1993744.1993753, June 2011, <https://dl.acm.org/doi/10.1145/1993744.1993753>.

[ARED01] Floyd, S., Gummadi, R., and S. Shenker, "Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management", ACIRI Technical Report 301, August 2001, <https://www.icsi.berkeley.edu/icsi/node/2032>.

[BBR-CC] Cardwell, N., Cheng, Y., Hassas Yeganeh, S., Swett, I., and V. Jacobson, "BBR Congestion Control", Work in Progress, Internet-Draft, draft-cardwell-iccrg-bbr-congestion-control-02, 7 March 2022, <https://datatracker.ietf.org/doc/html/draft-cardwell-iccrg-bbr-congestion-control-02>.

[BBRv2] "TCP BBR v2 Alpha/Preview Release", commit 17700ca, June

2022, <https://github.com/google/bbr>.

[Bufferbloat] The Bufferbloat community, "Bufferbloat", <https://bufferbloat.net/>.

[COBALT] Palmei, J., Gupta, S., Imputato, P., Morton, J., Tahiliani, M. P., Avallone, S., and D. Täht, "Design and Evaluation of COBALT Queue Discipline", IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN), DOI 10.1109/LANMAN.2019.8847054, July 2019, <https://ieeexplore.ieee.org/abstract/document/8847054>.

[CTCP] Sridharan, M., Tan, K., Bansal, D., and D. Thaler, "Compound TCP: A New TCP Congestion Control for High-Speed and Long Distance Networks", Work in Progress, Internet-Draft, draft-sridharan-tcpm-ctcp-02, 3 November 2008, <https://datatracker.ietf.org/doc/html/draft-sridharan-tcpm-ctcp-02>.

[DCttH19] De Schepper, K., Bondarenko, O., Tilmans, O., and B. Briscoe, "'Data Centre to the Home': Ultra-Low Latency for All", Updated RITE project Technical Report, July 2019, <https://bobbriscoe.net/projects/latency/dctth_journal_draft20190726.pdf>.

[DOCSIS-QPROT] Briscoe, B., Ed. and G. White, "The DOCSIS® Queue Protection Algorithm to Preserve Low Latency", Work in Progress, Internet-Draft, draft-briscoe-docsis-q-protection-06, 13 May 2022, <https://datatracker.ietf.org/doc/html/draft-briscoe-docsis-q-protection-06>.

[DualPI2Linux] Albisser, O., De Schepper, K., Briscoe, B., Tilmans, O., and H. Steen, "DUALPI2 - Low Latency, Low Loss and Scalable (L4S) AQM", Proceedings of Linux Netdev 0x13, March 2019, <https://www.netdevconf.org/0x13/session.html?talk-DUALPI2-AQM>.

[Dukkipati06] Dukkipati, N. and N. McKeown, "Why Flow-Completion Time is the Right Metric for Congestion Control", ACM SIGCOMM Computer Communication Review, Volume 36, Issue 1, pp. 59-62, DOI 10.1145/1111322.1111336, January 2006, <https://dl.acm.org/doi/10.1145/1111322.1111336>.

[ECN++] Bagnulo, M. and B. Briscoe, "ECN++: Adding Explicit Congestion Notification (ECN) to TCP Control Packets", Work in Progress, Internet-Draft, draft-ietf-tcpm-generalized-ecn-10, 27 July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-tcpm-generalized-ecn-10>.

[ECN-ENCAP]

Briscoe, B. and J. Kaippallimalil, "Guidelines for Adding Congestion Notification to Protocols that Encapsulate IP", Work in Progress, Internet-Draft, draft-ietf-tsvwg-ecn-encap-guidelines-17, 11 July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-tsvwg-ecn-encap-guidelines-17>.

[ecn-fallback] Briscoe, B. and A. Ahmed, "TCP Prague Fall-back on Detection of a Classic ECN AQM", Technical Report: TR-BB-2019-002, DOI 10.48550/arXiv.1911.00710, February 2021, <https://arxiv.org/abs/1911.00710>.

[ECN-SHIM] Briscoe, B., "Propagating Explicit Congestion Notification Across IP Tunnel Headers Separated by a Shim", Work in Progress, Internet-Draft, draft-ietf-tsvwg-rfc6040update-shim-15, 11 July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-tsvwg-rfc6040update-shim-15>.

[Heist21] "L4S Tests", commit e21cd91, August 2021, <https://github.com/heistp/l4s-tests>.

[L4S-DIFFSERV] Briscoe, B., "Interactions between Low Latency, Low Loss, Scalable Throughput (L4S) and Differentiated Services", Work in Progress, Internet-Draft, draft-briscoe-tsvwg-l4s-diffserv-02, 1 November 2018, <https://datatracker.ietf.org/doc/html/draft-briscoe-tsvwg-l4s-diffserv-02>.

[L4Seval22] De Schepper, K., Albisser, O., Tilmans, O., and B. Briscoe, "Dual Queue Coupled AQM: Deployable Very Low Queuing Delay for All", Preprint submitted to IEEE/ACM Transactions on Networking, DOI 10.48550/arXiv.2209.01078, September 2022, <https://arxiv.org/abs/2209.01078>.

[L4SOPS] White, G., Ed., "Operational Guidance for Deployment of L4S in the Internet", Work in Progress, Internet-Draft, draft-ietf-tsvwg-l4sops-03, 28 April 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-tsvwg-l4sops-03>.

[LinuxPacedChirping] Misund, J. and B. Briscoe, "Paced Chirping - Rethinking TCP start-up", Proceedings of Linux Netdev 0x13, March 2019, <https://legacy.netdevconf.info/0x13/session.html?talk-chirp>.

[NQB-PHB] White, G. and T. Fossati, "A Non-Queue-Building Per-Hop Behavior (NQB PHB) for Differentiated Services", Work in Progress, Internet-Draft, draft-ietf-tsvwg-nqb-15, 11 January 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-tsvwg-nqb-15>.

[PI2] De Schepper, K., Bondarenko, O., Tsang, I., and B. Briscoe, "PI^2: A Linearized AQM for both Classic and Scalable TCP", Proceedings of ACM CoNEXT 2016, pp. 105-119, DOI 10.1145/2999572.2999578, December 2016, <https://dl.acm.org/citation.cfm?doid=2999572.2999578>.

[PRAGUE-CC] De Schepper, K., Tilmans, O., and B. Briscoe, Ed., "Prague Congestion Control", Work in Progress, Internet-Draft, draft-briscoe-iccrg-prague-congestion-control-01, 11 July 2022, <https://datatracker.ietf.org/doc/html/draft-briscoe-iccrg-prague-congestion-control-01>.

[PragueLinux] Briscoe, B., De Schepper, K., Albisser, O., Misund, J., Tilmans, O., Kühlewind, M., and A. Ahmed, "Implementing the 'TCP Prague' Requirements for L4S", Proceedings of Linux Netdev 0x13, March 2019, <https://www.netdevconf.org/0x13/session.html?talk-tcp-prague-l4s>.

[QV] Briscoe, B. and P. Hurtig, "Report on Prototype Development and Evaluation of Network and Interaction Techniques", RITE Technical Report, Deliverable 2.3, Appendix C.2: "Up to Speed with Queue View", September 2015, <https://riteproject.files.wordpress.com/2015/12/rite-deliverable-2-3.pdf>.

[RELENTLESS] Mathis, M., "Relentless Congestion Control", Work in Progress, Internet-Draft, draft-mathis-iccrg-relentless-tcp-00, 4 March 2009, <https://datatracker.ietf.org/doc/html/draft-mathis-iccrg-relentless-tcp-00>.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <https://www.rfc-editor.org/info/rfc2474>.

[RFC3246] Davie, B., Charny, A., Bennet, J.C.R., Benson, K., Le Boudec, J.Y., 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>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/info/rfc4301>.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <https://www.rfc-editor.org/info/rfc4302>.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <https://www.rfc-editor.org/info/rfc4303>.

[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>.

[RFC4341] Floyd, S. and E. Kohler, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control", RFC 4341, DOI 10.17487/RFC4341, March 2006, <https://www.rfc-editor.org/info/rfc4341>.

[RFC4342] Floyd, S., Kohler, E., and J. Padhye, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC)", RFC 4342, DOI 10.17487/RFC4342, March 2006, <https://www.rfc-editor.org/info/rfc4342>.

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

[RFC5033] Floyd, S. and M. Allman, "Specifying New Congestion Control Algorithms", BCP 133, RFC 5033, DOI 10.17487/RFC5033, August 2007, <https://www.rfc-editor.org/info/rfc5033>.

[RFC5129] Davie, B., Briscoe, B., and J. Tay, "Explicit Congestion Marking in MPLS", RFC 5129, DOI 10.17487/RFC5129, January 2008, <https://www.rfc-editor.org/info/rfc5129>.

[RFC5348] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 5348, DOI 10.17487/RFC5348, September 2008, <https://www.rfc-editor.org/info/rfc5348>.

[RFC5562] Kuzmanovic, A., Mondal, A., Floyd, S., and K. Ramakrishnan, "Adding Explicit Congestion Notification (ECN) Capability to TCP's SYN/ACK Packets", RFC 5562, DOI 10.17487/RFC5562, June 2009, <https://www.rfc-editor.org/info/rfc5562>.

[RFC5622] Floyd, S. and E. Kohler, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion ID 4: TCP-Friendly Rate Control for Small Packets (TFRC-SP)", RFC 5622, DOI 10.17487/RFC5622, August 2009,

<https://www.rfc-editor.org/info/rfc5622>.

[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>.

[RFC5706] Harrington, D., "Guidelines for Considering Operations and Management of New Protocols and Protocol Extensions", RFC 5706, DOI 10.17487/RFC5706, November 2009, <https://www.rfc-editor.org/info/rfc5706>.

[RFC5865] Baker, F., Polk, J., and M. Dolly, "A Differentiated Services Code Point (DSCP) for Capacity-Admitted Traffic", RFC 5865, DOI 10.17487/RFC5865, May 2010, <https://www.rfc-editor.org/info/rfc5865>.

[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>.

[RFC6077] Papadimitriou, D., Ed., Welzl, M., Scharf, M., and B. Briscoe, "Open Research Issues in Internet Congestion Control", RFC 6077, DOI 10.17487/RFC6077, February 2011, <https://www.rfc-editor.org/info/rfc6077>.

[RFC6660] Briscoe, B., Moncaster, T., and M. Menth, "Encoding Three Pre-Congestion Notification (PCN) States in the IP Header Using a Single Diffserv Codepoint (DSCP)", RFC 6660, DOI 10.17487/RFC6660, July 2012, <https://www.rfc-editor.org/info/rfc6660>.

[RFC6675] Blanton, E., Allman, M., Wang, L., Jarvinen, I., Kojo, M., and Y. Nishida, "A Conservative Loss Recovery Algorithm Based on Selective Acknowledgment (SACK) for TCP", RFC 6675, DOI 10.17487/RFC6675, August 2012, <https://www.rfc-editor.org/info/rfc6675>.

[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>.

[RFC7567] Baker, F., Ed. and G. Fairhurst, Ed., "IETF Recommendations Regarding Active Queue Management", BCP 197, RFC 7567, DOI 10.17487/RFC7567, July 2015, <https://www.rfc-editor.org/info/rfc7567>.

[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>.

[RFC8083] Perkins, C. and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions", RFC 8083, DOI 10.17487/RFC8083, March 2017, <https://www.rfc-editor.org/info/rfc8083>.

[RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <https://www.rfc-editor.org/info/rfc8085>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[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>.

[RFC8888] Sarker, Z., Perkins, C., Singh, V., and M. Ramalho, "RTP Control Protocol (RTCP) Feedback for Congestion Control", RFC 8888, DOI 10.17487/RFC8888, January 2021, <https://www.rfc-editor.org/info/rfc8888>.

[RFC8985] Cheng, Y., Cardwell, N., Dukkipati, N., and P. Jha, "The

RACK-TLP Loss Detection Algorithm for TCP", RFC 8985, DOI 10.17487/RFC8985, February 2021, <https://www.rfc-editor.org/info/rfc8985>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[RFC9001] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <https://www.rfc-editor.org/info/rfc9001>.

[RFC9147] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/info/rfc9147>.

[RFC9330] Briscoe, B., Ed., De Schepper, K., Bagnulo, M., and G. White, "Low Latency, Low Loss, and Scalable Throughput (L4S) Internet Service: Architecture", RFC 9330, DOI 10.17487/RFC9330, January 2023, <https://www.rfc-editor.org/info/rfc9330>.

[RFC9332] De Schepper, K., Briscoe, B., Ed., and G. White, "Dual-Queue Coupled Active Queue Management (AQM) for Low Latency, Low Loss, and Scalable Throughput (L4S)", RFC 9332, DOI 10.17487/RFC9332, January 2023, <https://www.rfc-editor.org/info/rfc9332>.

[Savage-TCP]
Savage, S., Cardwell, N., Wetherall, D., and T. Anderson, "TCP Congestion Control with a Misbehaving Receiver", ACM SIGCOMM Computer Communication Review, Volume 29, Issue 5, pp. 71-78, DOI 10.1145/505696.505704, October 1999, <https://dl.acm.org/doi/abs/10.1145/505696.505704>.

[SCReAM-L4S]
"SCReAM", commit 140e292, November 2022, <https://github.com/EricssonResearch/scream>.

[SCTP-ECN] Stewart, R., Tüxen, M., and X. Dong, "ECN for Stream Control Transmission Protocol (SCTP)", Work in Progress, Internet-Draft, draft-stewart-tsvwg-sctpecn-05, 15 January 2014, <https://datatracker.ietf.org/doc/html/draft-stewart-tsvwg-sctpecn-05>.

[sub-mss-prob]
Briscoe, B. and K. De Schepper, "Scaling TCP's Congestion Window for Small Round Trip Times", BT Technical Report: TR-TUB8-2015-002, DOI 10.48550/arXiv.1904.07598, May 2015, <https://arxiv.org/abs/1904.07598>.

[TCP-CA] Jacobson, V. and M. J. Karels, "Congestion Avoidance and Control", Laurence Berkeley Labs Technical Report, November 1988, <https://ee.lbl.gov/papers/congavoid.pdf>.

[TCPPrague]
Briscoe, B., "Notes: DCTCP evolution 'bar BoF': Tue 21 Jul 2015, 17:40, Prague", message to the tcpPrague mailing list, July 2015, <https://www.ietf.org/mail-archive/web/tcpprague/current/msg00001.html>.

[TRILL-ECN-SUPPORT]
Eastlake 3rd, D. and B. Briscoe, "TRILL (TRansparent Interconnection of Lots of Links): ECN (Explicit Congestion Notification) Support", Work in Progress, Internet-Draft, draft-ietf-trill-ecn-support-07, 25 February 2018, <https://datatracker.ietf.org/doc/html/draft-ietf-trill-ecn-support-07>.

[VCP] Xia, Y., Subramanian, L., Stoica, I., and S. Kalyanaraman, "One more bit is enough", SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 37-48, DOI 10.1145/1080091.1080098, August 2005, <https://doi.acm.org/10.1145/1080091.1080098>.

Appendix A. Rationale for the 'Prague L4S Requirements'

This appendix is informative, not normative. It gives a list of modifications to current Scalable congestion controls so that they can be deployed over the public Internet and coexist safely with existing traffic. The list complements the normative requirements in Section 4 that a sender has to comply with before it can set the L4S identifier in packets it sends into the Internet. As well as rationale for safety improvements (the requirements in Section 4), this appendix also includes preferable performance improvements (optimizations).

The requirements and recommendations in Section 4 have become known as the 'Prague L4S Requirements', because they were originally identified at an ad hoc meeting during IETF 94 in Prague [TCPPrague]. They were originally called the 'TCP Prague Requirements', but they are not solely applicable to TCP, so the name and wording has been generalized for all transport protocols, and the name 'TCP Prague' is now used for a specific implementation of the requirements.

At the time of writing, DCTCP [RFC8257] is the most widely used Scalable transport protocol. In its current form, DCTCP is specified to be deployable only in controlled environments. Deploying it in the public Internet would lead to a number of issues, from both the safety and the performance perspective. The modifications and additional mechanisms listed in this section will be necessary for its deployment over the global Internet. Where an example is needed, DCTCP is used as a base, but the requirements in Section 4 apply equally to other Scalable congestion controls, covering adaptive real-time media, etc., not just capacity-seeking behaviours.

A.1. Rationale for the Requirements for Scalable Transport Protocols

A.1.1. Use of L4S Packet Identifier

Description: A Scalable congestion control needs to distinguish the packets it sends from those sent by Classic congestion controls (see the precise normative requirement wording in Section 4.1).

Motivation: It needs to be possible for a network node to classify L4S packets without flow state into a queue that applies an L4S ECN-marking behaviour and isolates L4S packets from the queuing delay of Classic packets.

A.1.2. Accurate ECN Feedback

Description: The transport protocol for a Scalable congestion control needs to provide timely, accurate feedback about the extent of ECN marking experienced by all packets (see the precise normative requirement wording in Section 4.2).

Motivation: Classic congestion controls only need feedback about the existence of a congestion episode within a round trip, not precisely how many packets were ECN-marked or dropped. Therefore, in 2001, when ECN feedback was added to TCP [RFC3168], it could not inform the sender of more than one ECN mark per RTT. Since then, requirements for more accurate ECN feedback in TCP have been defined in [RFC7560], and [ACCECN] specifies a change to the TCP protocol to satisfy these requirements. Most other transport protocols already satisfy this requirement (see Section 4.2).

A.1.3. Capable of Replacement by Classic Congestion Control

Description: It needs to be possible to replace the implementation of a Scalable congestion control with a Classic control (see the precise normative requirement wording in Section 4.3, Paragraph 8, Item 1).

Motivation: L4S is an experimental protocol; therefore, it seems prudent to be able to disable it at source in case of insurmountable problems, perhaps due to some unexpected interaction on a particular sender; over a particular path or network; or with a particular receiver, or even ultimately an insurmountable problem with the experiment as a whole.

A.1.4. Fall Back to Classic Congestion Control on Packet Loss

Description: As well as responding to ECN markings in a scalable way, a Scalable congestion control needs to react to packet loss in a way that will coexist safely with a Reno congestion control [RFC5681] (see the precise normative requirement wording in Section 4.3, Paragraph 8, Item 2).

Motivation: Part of the safety conditions for deploying a Scalable congestion control on the public Internet is to make sure that it behaves properly when it builds a queue at a network bottleneck that has not been upgraded to support L4S. Packet loss can have many causes, but it usually has to be conservatively assumed that it is a sign of congestion. Therefore, on detecting packet loss, a Scalable congestion control will need to fall back to Classic congestion control behaviour. If it does not comply, it could starve Classic traffic.

A Scalable congestion control can be used for different types of transport, e.g., for real-time media or for reliable transport like TCP. Therefore, the particular Classic congestion control behaviour to fall back on will need to be dependent on the specific congestion control implementation. In the particular case of DCTCP, the DCTCP specification [RFC8257] states that "A DCTCP sender MUST react to loss episodes in the same way as conventional TCP,...". To ensure any Scalable congestion control is safe to deploy over the public Internet, Item 2 of Section 4.3 in the present spec does not require precisely the same response as Reno TCP, but it does require a response that will coexist safely with Classic congestion controls like Reno.

Even though a bottleneck is L4S capable, it might still become overloaded and have to drop packets. In this case, the sender may receive a high proportion of packets marked with the CE codepoint and also experience loss. Current DCTCP implementations each react differently to this situation. One approach is to react only to the drop signal (e.g., by halving the cwnd); another approach is to react to both signals, which reduces cwnd by more than half. A compromise between these two has been proposed where the loss response is adjusted to result in a halving when combined with any ECN response earlier in the same round. We believe that further experimentation is needed to understand what is the best behaviour for the public Internet, which may or may not be one of these existing approaches.

A.1.5. Coexistence with Classic Congestion Control at Classic ECN Bottlenecks

Description: Monitoring has to be in place so that a non-L4S but ECN-capable AQM can be detected at path bottlenecks. This is in case such an AQM has been implemented in a shared queue, in which case any long-running Scalable flow would predominate over any simultaneous long-running Classic flow sharing the queue. The precise requirement wording in Section 4.3, Paragraph 8, Item 3 is written so that such a problem could be resolved either in real time or via administrative intervention.

Motivation: Similarly to the discussion in Appendix A.1.4, this requirement in Section 4.3 is a safety condition to ensure an L4S congestion control coexists well with Classic flows when it builds a queue at a shared network bottleneck that has not been upgraded to support L4S. Nonetheless, if necessary, it is considered reasonable to resolve such problems over management timescales (possibly involving human intervention) because:

* although a Classic flow can considerably reduce its throughput in the face of a competing Scalable flow, it still makes progress and does not starve;

* implementations of a Classic ECN AQM in a queue that is intended to be shared are believed to be rare; and

* detection of such AQMs is not always clear-cut; so focused out-of-

band testing (or even contacting the relevant network operator) would improve certainty.

The relevant normative requirement (Section 4.3) is therefore divided into three stages: monitoring, detection, and action:

Monitoring: Monitoring involves collection of the measurement data to be analysed. Monitoring is expressed as a "MUST" for uncontrolled environments, although the placement of the monitoring function is left open. Whether monitoring has to be applied in real time is expressed as a "SHOULD". This allows for the possibility that the operator of an L4S sender (e.g., a Content Distribution Network (CDN)) might prefer to test out-of-band for signs of Classic ECN AQMs, perhaps to avoid continually consuming resources to monitor live traffic.

Detection: Detection involves analysis of the monitored data to detect the likelihood of a Classic ECN AQM. Detection can either directly detect actual coexistence problems between flows or aim to identify AQM technologies that are likely to present coexistence problems, based on knowledge of AQMs deployed at the time. The requirements recommend that detection occurs live in real time. However, detection is allowed to be deferred (e.g., it might involve further testing targeted at candidate AQMs).

Action: This involves the act of switching the sender to a Classic congestion control. This might occur in real time within the congestion control for the subsequent duration of a flow, or it might involve administrative action to switch to Classic congestion control for a specific interface or for a certain set of destination addresses.

Instead of the sender taking action itself, the operator of the sender (e.g., a CDN) might prefer to ask the network operator to modify the Classic AQM's treatment of L4S packets; ensure L4S packets bypass the AQM; or upgrade the AQM to support L4S (see the L4S operational guidance [L4SOPS]). If L4S flows then no longer shared the Classic ECN AQM, they would obviously no longer detect it, and the requirement to act on it would no longer apply.

The whole set of normative requirements concerning Classic ECN AQMs in Section 4.3 is worded so that it does not apply in controlled environments, such as private networks or data-centre networks. CDN servers placed within an access ISP's network can be considered as a single controlled environment, but any onward networks served by the access network, including all the attached customer networks, would be unlikely to fall under the same degree of coordinated control. Monitoring is expressed as a "MUST" for these uncontrolled segments of paths (e.g., beyond the access ISP in a home network), because there is a possibility that there might be a shared queue Classic ECN AQM in that segment. Nonetheless, the intent of the wording is to only require occasional monitoring of these uncontrolled regions and not to burden CDN operators if monitoring never uncovers any potential problems.

More detailed discussion of all the above options and alternatives can be found in the L4S operational guidance [L4SOPS].

Having said all the above, the approach recommended in Section 4.3 is to monitor, detect, and act in real time on live traffic. A passive monitoring algorithm to detect a Classic ECN AQM at the bottleneck and fall back to Classic congestion control is described in an extensive technical report [ecn-fallback], which also provides a link to Linux source code and a large online visualization of its evaluation results. Very briefly, the algorithm primarily monitors RTT variation using the same algorithm that maintains the mean deviation of TCP's smoothed RTT, but it smooths over a duration of the order of a Classic sawtooth. The outcome is also conditioned on other metrics such as the presence of CE marking and congestion avoidance phase having stabilized. The report also identifies further work to improve the approach, for instance, improvements with low-capacity links and combining the measurements with a cache of what had been learned about a path in previous connections. The report also suggests alternative approaches.

Although using passive measurements within live traffic (as above) can detect a Classic ECN AQM, it is much harder (perhaps impossible) to determine whether or not the AQM is in a shared queue. Nonetheless, this is much easier using active test traffic out-of-band because two flows can be used. Section 4 of the same report [ecn-fallback] describes a simple technique to detect a Classic ECN AQM and determine whether it is in a shared queue, which is summarized here.

An L4S-enabled test server could be set up so that, when a test client accesses it, it serves a script that gets the client to open two parallel long-running flows. It could serve one with a Classic congestion control (C, that sets ECT(0)) and one with a Scalable CC (L, that sets ECT(1)). If neither flow induces any ECN marks, it can be presumed that the path does not contain a Classic ECN AQM. If either flow induces some ECN marks, the server could measure the relative flow rates and round-trip times of the two flows. Table 2 shows the AQM that can be inferred for various cases (presuming no more types of AQM behaviour than those known at the time of writing).

Rate	RTT	Inferred AQM
L > C	L = C	Classic ECN AQM (FIFO)
L = C	L = C	Classic ECN AQM (FQ)
L = C	L < C	FQ-L4S AQM
L ~= C	L < C	DualQ Coupled AQM
L = L4S; C = Classic		

Table 2: Out-of-Band Testing with Two Parallel Flows

Finally, we motivate the recommendation in Section 4.3 that a Scalable congestion control is not expected to change to setting ECT(0) while it adapts its behaviour to coexist with Classic flows. This is because the sender needs to continue to check whether it made the right decision and switch back if it was wrong, or if a different link becomes the bottleneck:

* If, as recommended, the sender changes only its behaviour but not its codepoint to Classic, its codepoint will still be compatible with either an L4S or a Classic AQM. If the bottleneck does actually support both, it will still classify ECT(1) into the same L4S queue, where the sender can measure that switching to Classic behaviour was wrong so that it can switch back.

* In contrast, if the sender changes both its behaviour and its codepoint to Classic, even if the bottleneck supports both, it will classify ECT(0) into the Classic queue, reinforcing the sender's incorrect decision so that it never switches back.

* Also, not changing its codepoint avoids the risk of being flipped to a different path by a load balancer or multipath routing that hashes on the whole of the former Type-of-Service (ToS) byte (which is unfortunately still a common pathology).

> Note that if a flow is configured to _only_ use a Classic congestion control, it is then entirely appropriate not to use ECT(1).

A.1.6. Reduce RTT Dependence

Description: A Scalable congestion control needs to reduce RTT bias as much as possible at least over the low-to-typical range of RTTs that will interact in the intended deployment scenario (see the precise normative requirement wording in Section 4.3, Paragraph 8, Item 4).

Motivation: The throughput of Classic congestion controls is known to be inversely proportional to RTT, so one would expect flows over very low RTT paths to nearly starve flows over larger RTTs. However, Classic congestion controls have never allowed a very low RTT path to exist because they induce a large queue. For instance, consider two paths with base RTT 1 ms and 100 ms. If a Classic congestion control induces a 100 ms queue, it turns these RTTs into 101 ms and 200 ms, leading to a throughput ratio of about 2:1. Whereas if a Scalable congestion control induces only a 1 ms queue, the ratio is 2:101, leading to a throughput ratio of about 50:1.

Therefore, with very small queues, long RTT flows will essentially starve, unless Scalable congestion controls comply with the requirement in Section 4.3.

Over higher than typical RTTs, L4S flows can use the same RTT bias as in current Classic congestion controls and still work satisfactorily. So there is no additional requirement in Section 4.3 for high RTT L4S flows to remove RTT bias -- they can, but they don't have to.

One way for a Scalable congestion control to satisfy these requirements is to make its additive increase behave as if it were a standard Reno flow but over a larger RTT by using a virtual RTT (rtt_virt) that is a function of the actual RTT (rtt). Example functions might be:

```
rtt_virt = max(rtt, 25 ms)

rtt_virt = rtt + 10 ms
```

These example functions are chosen so that, as the actual RTT reduces from high to low, the virtual RTT reduces less (see [PRAGUE-CC] for details).

However, short RTT flows can more rapidly respond to changes in available capacity, whether due to other flows arriving and departing or radio capacity varying. So it would be wrong to require short RTT flows to be as sluggish as long RTT flows, which would unnecessarily underutilize capacity and result in unnecessary overshoots and undershoots (instability). Therefore, rather than requiring strict RTT independence, the wording in Item 4 of Section 4.3 is "as independent of RTT as possible without compromising stability or utilization". This allows shorter RTT flows to exploit their agility advantage.

A.1.7. Scaling Down to Fractional Congestion Windows

Description: A Scalable congestion control needs to remain responsive to congestion when typical RTTs over the public Internet are significantly smaller because they are no longer inflated by queuing delay (see the precise normative requirement wording in Section 4.3, Paragraph 8, Item 5).

Motivation: As currently specified, the minimum congestion window of ECN-capable TCP (and its derivatives) is expected to be 2 sender maximum segment sizes (SMSS), or 1 SMSS after a retransmission timeout. Once the congestion window reaches this minimum, if there is further ECN marking, TCP is meant to wait for a retransmission timeout before sending another segment (see Section 6.1.2 of the ECN spec [RFC3168]). In practice, most known window-based congestion control algorithms become unresponsive to ECN congestion signals at this point. No matter how much ECN marking, the congestion window no longer reduces. Instead, the sender's lack of any further congestion response forces the queue to grow, overriding any AQM and increasing queuing delay (making the window large enough to become responsive again). This can result in a stable but deeper queue, or it might drive the queue to loss, in which case the retransmission timeout mechanism acts as a backstop.

Most window-based congestion controls for other transport protocols have a similar minimum window, albeit when measured in bytes for those that use smaller packets.

L4S mechanisms significantly reduce queuing delay so, over the same path, the RTT becomes lower. Then, this problem becomes surprisingly common [sub-mss-prob]. This is because, for the same link capacity, smaller RTT implies a smaller window. For instance, consider a residential setting with an upstream broadband Internet access of 8 Mb/s, assuming a max segment size of 1500 B. Two upstream flows will each have the minimum window of 2 SMSS if the RTT is 6 ms or less, which is quite common when accessing a nearby data centre. So any more than two such parallel TCP flows will become unresponsive to ECN and increase queuing delay.

Unless Scalable congestion controls address the requirement in Section 4.3 from the start, they will frequently become unresponsive to ECN, negating the low-latency benefit of L4S, for themselves and for others.

That would seem to imply that Scalable congestion controllers ought to be required to be able work with a congestion window less than 1 SMSS. For instance, if an ECN-capable TCP gets an ECN mark when it is already sitting at a window of 1 SMSS, [RFC3168] requires it to defer sending for a retransmission timeout. A less drastic but more complex mechanism can maintain a congestion window less than 1 SMSS (significantly less if necessary), as described in [Ahmed19]. Other approaches are likely to be feasible.

However, the requirement in Section 4.3 is worded as a "SHOULD" because it is believed that the existence of a minimum window is not all bad. When competing with an unresponsive flow, a minimum window naturally protects the flow from starvation by at least keeping some data flowing.

By stating the requirement to go lower than 1 SMSS as a "SHOULD", while the requirement in [RFC3168] still stands as well, we shall be able to watch the choices of minimum window evolve in different Scalable congestion controllers.

A.1.8. Measuring Reordering Tolerance in Time Units

Description: When detecting loss, a Scalable congestion control needs to be tolerant to reordering over an adaptive time interval, which scales with throughput, rather than counting only in fixed units of packets, which does not scale (see the precise normative requirement wording in Section 4.3, Paragraph 8, Item 6).

Motivation: A primary purpose of L4S is scalable throughput (it's in the name). Scalability in all dimensions is, of course, also a goal of all IETF technology. The inverse linear congestion response in Section 4.3 is necessary, but not sufficient, to solve the congestion control scalability problem identified in [RFC3649]. As well as maintaining frequent ECN signals as rate scales, it is also important to ensure that a potentially false perception of loss does not limit throughput scaling.

End systems cannot know whether a missing packet is due to loss or reordering, except in hindsight -- if it appears later. So they can only deem that there has been a loss if a gap in the sequence space has not been filled, either after a certain number of subsequent packets has arrived (e.g., the 3 DupACK rule of standard TCP congestion control [RFC5681]) or after a certain amount of time (e.g., the RACK approach [RFC8985]).

As we attempt to scale packet rate over the years:

* Even if only _some_ sending hosts still deem that loss has occurred by counting reordered packets, _all_ networks will have to keep reducing the time over which they keep packets in order. If some link technologies keep the time within which reordering occurs roughly unchanged, then loss over these links, as perceived by these hosts, will appear to continually rise over the years.

* In contrast, if all senders detect loss in units of time, the time over which the network has to keep packets in order stays roughly invariant.

Therefore, hosts have an incentive to detect loss in time units (so as not to fool themselves too often into detecting losses when there are none). And for hosts that are changing their congestion control implementation to L4S, there is no downside to including time-based loss detection code in the change (loss recovery implemented in hardware is an exception, which is covered later). Therefore, requiring L4S hosts to detect loss in time-based units would not be a burden.

If the requirement in Section 4.3 were not placed on L4S hosts, even though it would be no burden on hosts to comply, all networks would face unnecessary uncertainty over whether some L4S hosts might be detecting loss by counting packets. Then, _all_ link technologies would have to unnecessarily keep reducing the time within which reordering occurs. That is not a problem for some link technologies, but it becomes increasingly challenging for other link technologies to continue to scale, particularly those relying on channel bonding for scaling, such as LTE, 5G, and Data Over Cable Service Interface Specification (DOCSIS).

Given Internet paths traverse many link technologies, any scaling limit for these more challenging access link technologies would become a scaling limit for the Internet as a whole.

It might be asked how it helps to place this loss detection requirement only on L4S hosts, because networks will still face uncertainty over whether non-L4S flows are detecting loss by counting DupACKs. The answer is that those link technologies for which it is challenging to keep squeezing the reordering time will only need to do so for non-L4S traffic (which they can do because the L4S identifier is visible at the IP layer). Therefore, they can focus their processing and memory resources into scaling non-L4S (Classic) traffic. Then, the higher the proportion of L4S traffic, the less of a scaling challenge they will have.

To summarize, there is no reason for L4S hosts not to be part of the solution instead of part of the problem.

Requirement ("MUST") or recommendation ("SHOULD")? As explained above, this is a subtle interoperability issue between hosts and networks, which seems to need a "MUST". Unless networks can be certain that all L4S hosts follow the time-based approach, they still have to cater for the worst case -- continually squeeze reordering into a smaller and smaller duration -- just for hosts that might be using the counting approach. However, it was decided to express this as a recommendation, using "SHOULD". The main justification was that networks can still be fairly certain that L4S hosts will follow this recommendation, because following it offers only gain and no pain.

Details:

The time spent recovering a loss is much more significant for short flows than long; therefore, a good compromise is to adapt the reordering window from a small fraction of the RTT at the start of a flow to a larger fraction of the RTT for flows that continue for many round trips.

This is broadly the approach adopted by RACK [RFC8985]. However, RACK starts with the 3 DupACK approach, because the RTT estimate is not necessarily stable. As long as the initial window is paced, such initial use of 3 DupACK counting would amount to time-based loss detection and therefore would satisfy the time-based loss detection recommendation of Section 4.3. This is because pacing of the initial window would ensure that 3 DupACKs early in the connection would be spread over a small fraction of the round trip.

As mentioned above, hardware implementations of loss recovery using DupACK counting exist (e.g., some implementations of Remote Direct Memory Access over Converged Ethernet version 2 (RoCEv2)). For low latency, these implementations can change their congestion control to implement L4S, because the congestion control (as distinct from loss recovery) is implemented in software. But they cannot easily satisfy this loss recovery requirement. However, it is believed they do not need to, because such implementations are believed to solely exist in controlled environments, where the network technology keeps reordering extremely low anyway. This is why controlled environments with hardly any reordering are excluded from the scope of the normative recommendation in Section 4.3.

Detecting loss in time units also prevents the ACK-splitting attacks described in [Savage-TCP].

A.2. Scalable Transport Protocol Optimizations

A.2.1. Setting ECT in Control Packets and Retransmissions

Description: This item concerns TCP and its derivatives (e.g., SCTP) as well as RTP/RTCP [RFC6679]. The original specification of ECN for TCP precluded the use of ECN on control packets and retransmissions. Similarly, [RFC6679] precludes the use of ECT on RTCP datagrams, in case the path changes after it has been checked for ECN traversal. To improve performance, Scalable transport protocols ought to enable ECN at the IP layer in TCP control packets (SYN, SYN-ACK, pure ACKs, etc.) and in retransmitted packets. The same is true for other transports, e.g., SCTP and RTCP.

Motivation (TCP): [RFC3168] prohibits the use of ECN on these types of TCP packets, based on a number of arguments. This means these packets are not protected from congestion loss by ECN, which considerably harms performance, particularly for short flows. ECN++ [ECN++] proposes experimental use of ECN on all types of TCP packets as long as AccECN feedback [ACCECN] is available (which itself satisfies the accurate feedback requirement in Section 4.2 for using a Scalable congestion control).

Motivation (RTCP): L4S experiments in general will need to observe the rule in the RTP ECN spec [RFC6679] that precludes ECT on RTCP datagrams. Nonetheless, as ECN usage becomes more widespread, it would be useful to conduct specific experiments with ECN-capable RTCP to gather data on whether such caution is necessary.

A.2.2. Faster than Additive Increase

Description: It would improve performance if Scalable congestion controls did not limit their congestion window increase to the standard additive increase of 1 SMSS per round trip [RFC5681] during congestion avoidance. The same is true for derivatives of TCP congestion control, including similar approaches used for real-time media.

Motivation: As currently defined [RFC8257], DCTCP uses the conventional Reno additive increase in the congestion avoidance phase. When the available capacity suddenly increases (e.g., when another flow finishes or if radio capacity increases) it can take very many round trips to take advantage of the new capacity. TCP CUBIC [RFC8312] was designed to solve this problem, but as flow rates have continued to increase, the delay accelerating into available capacity has become prohibitive. See, for instance, the examples in Section 5.1 of the L4S architecture [RFC9330]. Even when out of its Reno-friendly mode, every 8 times scaling of CUBIC's flow rate leads to 2 times more acceleration delay.

In the steady state, DCTCP induces about 2 ECN marks per round trip, so it is possible to quickly detect when these signals have disappeared and seek available capacity more rapidly, while minimizing the impact on other flows (Classic and Scalable) [LinuxPacedChirping]. Alternatively, approaches such as Adaptive-Acceleration Data Center TCP (A2DTCP) [A2DTCP]) have been proposed to address this problem in data centres, which might be deployable over the public Internet.

A.2.3. Faster Convergence at Flow Start

Description: It would improve performance if Scalable congestion controls converged (reached their steady-state share of the capacity) faster than Classic congestion controls or at least no slower. This affects the flow start behaviour of any L4S congestion control derived from a Classic transport that uses TCP slow start, including those for real-time media.

Motivation: As an example, a new DCTCP flow takes longer than a Classic congestion control to obtain its share of the capacity of the bottleneck when there are already ongoing flows using the bottleneck capacity. In a data-centre environment, DCTCP takes about 1.5 to 2 times longer to converge due to the much higher typical level of ECN marking that DCTCP background traffic induces, which causes new flows to exit slow start early [Alizadeh-stability]. In testing for use over the public Internet, the convergence time of DCTCP relative to a regular loss-based TCP slow start is even less favourable [LinuxPacedChirping] due to the shallow ECN-marking threshold needed for L4S. It is exacerbated by the typically greater mismatch between the link rate of the sending host and typical Internet access bottlenecks. This problem is detrimental in general but would particularly harm the performance of short flows relative to Classic congestion controls.

Appendix B. Compromises in the Choice of L4S Identifier

This appendix is informative, not normative. As explained in Section 3, there is insufficient space in the IP header (v4 or v6) to fully accommodate every requirement. So the choice of L4S identifier involves trade-offs. This appendix records the pros and cons of the choice that was made.

Non-normative recap of the chosen codepoint scheme:

> Packets with ECT(1) and conditionally packets with CE signify L4S semantics as an alternative to the semantics of Classic ECN [RFC3168], specifically:
>
> - The ECT(1) codepoint signifies that the packet was sent by an L4S-capable sender.
>
> - Given the shortage of codepoints, both the L4S and Classic ECN sides of an AQM have to use the same CE codepoint to indicate that a packet has experienced congestion. If a packet that had already been marked CE in an upstream buffer arrived at a subsequent AQM, this AQM would then have to guess whether to classify CE packets as L4S or Classic ECN. Choosing the L4S treatment is a safer choice, because then a few Classic packets might arrive early rather than a few L4S packets arriving late.
>
> - Additional information might be available if the classifier were transport-aware. Then, it could classify a CE packet for Classic ECN treatment if the most recent ECT packet in the same flow had been set to ECT(0). However, the L4S service ought not need transport-layer awareness.

Cons:

Consumes the last ECN codepoint: The L4S service could potentially supersede the service provided by Classic ECN; therefore, using ECT(1) to identify L4S packets could ultimately mean that the ECT(0) codepoint was 'wasted' purely to distinguish one form of ECN from its successor.

ECN hard in some lower layers: It is not always possible to support the equivalent of an IP-ECN field in an AQM acting in a buffer below the IP layer [ECN-ENCAP]. Then, depending on the lower-layer scheme, the L4S service might have to drop rather than mark frames even though they might encapsulate an ECN-capable packet.

Risk of reordering Classic CE packets within a flow: Classifying all CE packets into the L4S queue risks any CE packets that were originally ECT(0) being incorrectly classified as L4S. If there were delay in the Classic queue, these incorrectly classified CE packets would arrive early, which is a form of reordering. Reordering within a microflow can cause TCP senders (and senders of similar transports) to retransmit spuriously. However, the risk of spurious retransmissions would be extremely low for the following reasons:

1. It is quite unusual to experience queuing at more than one bottleneck on the same path (the available capacities have to be identical).

2. In only a subset of these unusual cases would the first bottleneck support Classic ECN marking and the second L4S ECN marking. This would be the only scenario where some ECT(0) packets could be CE marked by an AQM supporting Classic ECN while subsequently the remaining ECT(0) packets would experience further delay through the Classic side of a subsequent L4S DualQ AQM.

3. Even then, when a few packets are delivered early, it takes very unusual conditions to cause a spurious retransmission, in contrast to when some packets are delivered late. The first bottleneck has to apply CE marks to at least N contiguous packets, and the second bottleneck has to inject an uninterrupted sequence of at least N of these packets between two packets earlier in the stream (where N is the reordering window that the transport protocol allows before it considers a packet is lost).

   For example, consider N=3, and consider the sequence of packets 100, 101, 102, 103,... Imagine that packets 150, 151, 152 from later in the flow are injected as follows: 100, 150, 151, 101, 152, 102, 103,... If this were late reordering, even one packet arriving out of sequence would trigger a spurious retransmission, but there is no spurious retransmission here with early reordering, because packet 101 moves the cumulative ACK counter forward before 3 packets have arrived out of order. Later, when packets 148, 149, 153,... arrive, even though there is a 3-packet hole, there will be no problem, because the packets to fill the hole are already in the receive buffer.

4. Even with the current TCP recommendation of N=3 [RFC5681], spurious retransmissions will be unlikely for all the above reasons. As RACK [RFC8985] is becoming widely deployed, it tends to adapt its reordering window to a larger value of N, which will make the chance of a contiguous sequence of N early arrivals vanishingly small.

5. Even a run of 2 CE marks within a Classic ECN flow is

unlikely, given FQ-CoDel is the only known widely deployed AQM that supports Classic ECN marking, and it takes great care to separate out flows and to space any markings evenly along each flow.

It is extremely unlikely that the above set of 5 eventualities that are each unusual in themselves would all happen simultaneously. But, even if they did, the consequences would hardly be dire: the odd spurious fast retransmission. Whenever the traffic source (a Classic congestion control) mistakes the reordering of a string of CE marks for a loss, one might think that it will reduce its congestion window as well as emitting a spurious retransmission. However, it would have already reduced its congestion window when the CE markings arrived early. If it is using ABE [RFC8511], it might reduce cwnd a little more for a loss than for a CE mark. But it will revert that reduction once it detects that the retransmission was spurious.

In conclusion, the impact of early reordering on spurious retransmissions due to CE being ambiguous will generally be vanishingly small.

Insufficient anti-replay window in some pre-existing VPNs: If delay is reduced for a subset of the flows within a VPN, the anti-replay feature of some VPNs is known to potentially mistake the difference in delay for a replay attack. Section 6.2 recommends that the anti-replay window at the VPN egress is sufficiently sized, as required by the relevant specifications. However, in some VPN implementations, the maximum anti-replay window is insufficient to cater for a large delay difference at prevailing packet rates. Section 6.2 suggests alternative work-rounds for such cases, but end users using L4S over a VPN will need to be able to recognize the symptoms of this problem, in order to seek out these work-rounds.

Hard to distinguish Classic ECN AQM: With this scheme, when a source receives ECN feedback, it is not explicitly clear which type of AQM generated the CE markings. This is not a problem for Classic ECN sources that send ECT(0) packets, because an L4S AQM will recognize the ECT(0) packets as Classic and apply the appropriate Classic ECN-marking behaviour.

However, in the absence of explicit disambiguation of the CE markings, an L4S source needs to use heuristic techniques to work out which type of congestion response to apply (see Appendix A.1.5). Otherwise, if long-running Classic flows are sharing a Classic ECN AQM bottleneck with long-running L4S flows, and the L4S flows apply an L4S response to the Classic CE signals, they would then outcompete the Classic flows. Experiments have shown that L4S flows can take about 20 times more capacity share than equivalent Classic flows. Nonetheless, as link capacity reduces (e.g., to 4 Mb/s), the inequality reduces. So Classic flows always make progress and are not starved.

When L4S was first proposed (in 2015, 14 years after the Classic ECN spec [RFC3168] was published), it was believed that Classic

ECN AQMs had failed to be deployed because research measurements had found little or no evidence of CE marking. In subsequent years, Classic ECN was included in FQ deployments; however, an FQ scheduler stops an L4S flow outcompeting Classic, because it enforces equality between flow rates. It is not known whether there have been any non-FQ deployments of Classic ECN AQMs in the subsequent years or whether there will be any in future.

An algorithm for detecting a Classic ECN AQM as soon as a flow stabilizes after start-up has been proposed [ecn-fallback] (see Appendix A.1.5 for a brief summary). Testbed evaluations of v2 of the algorithm have shown detection is reasonably good for Classic ECN AQMs, in a wide range of circumstances. However, although it can correctly detect an L4S ECN AQM in many circumstances, it is often incorrect at low link capacities and/or high RTTs. Although this is the safe way round, there is a danger that it will discourage use of the algorithm.

Non-L4S service for control packets: Solely for the case of TCP, the Classic ECN RFCs [RFC3168] and [RFC5562] require a sender to clear the IP-ECN field to Not-ECT on retransmissions and on certain control packets, specifically pure ACKs, window probes, and SYNs. When L4S packets are classified by the IP-ECN field, these TCP control packets would not be classified into an L4S queue and could therefore be delayed relative to the other packets in the flow. This would not cause reordering (because retransmissions are already out of order, and these control packets typically carry no data). However, it would make critical TCP control packets more vulnerable to loss and delay. To address this problem, ECN++ [ECN++] proposes an experiment in which all TCP control packets and retransmissions are ECN-capable as long as appropriate ECN feedback is available in each case.

Pros:

Should work end to end: The IP-ECN field generally propagates end to end across the Internet without being wiped or mangled, at least over fixed networks. Unlike the DSCP, the setting of the ECN field is at least meant to be forwarded unchanged by networks that do not support ECN.

Should work in tunnels: The L4S identifiers work across and within any tunnel that propagates the IP-ECN field in any of the variant ways it has been defined since ECN-tunneling was first specified in the year 2001 [RFC3168]. However, it is likely that some tunnels still do not implement ECN propagation at all.

Should work for many link technologies: At most, but not all, path bottlenecks there is IP awareness, so that L4S AQMs can be located where the IP-ECN field can be manipulated. Bottlenecks at lower-layer nodes without IP awareness have to either use drop to signal congestion or have a specific congestion notification facility defined for that link technology, including propagation to and from IP-ECN. The programme to define these is progressing, and in each case so far, the scheme already defined for ECN inherently supports L4S as well (see Section 6.1).

Could migrate to one codepoint: If all Classic ECN senders eventually evolve to use the L4S service, the ECT(0) codepoint could be reused for some future purpose but only once use of ECT(0) packets has reduced to zero, or near zero, which might never happen.

L4 not required: Being based on the IP-ECN field, this scheme does not need the network to access transport-layer flow IDs. Nonetheless, it does not preclude solutions that do.

Appendix C. Potential Competing Uses for the ECT(1) Codepoint

The ECT(1) codepoint of the IP-ECN field has already been assigned once for the ECN nonce spec [RFC3540], which has now been categorized as Historic [RFC8311]. ECN is probably the only remaining field in the Internet Protocol that is common to IPv4 and IPv6 and still has potential to work end to end, with tunnels and with lower layers. Therefore, ECT(1) should not be reassigned to a different experimental use (L4S) without carefully assessing competing potential uses. These fall into the categories described below.

C.1. Integrity of Congestion Feedback

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).

The Historic ECN nonce spec [RFC3540] proposed that a TCP sender could set either ECT(0) or ECT(1) in each packet of a flow and remember the sequence it had set. If any packet was lost or congestion marked, the receiver would miss that bit of the sequence. An ECN nonce receiver had to feed back the least-significant bit of the sum, so it could not suppress feedback of a loss or mark without a 50-50 chance of guessing the sum incorrectly.

It is highly unlikely that ECT(1) will be needed as a nonce for integrity protection of congestion notifications in future. The ECN nonce spec [RFC3540] has been reclassified as Historic, partly because other ways (that do not consume a codepoint in the IP header) have been developed to protect feedback integrity of TCP and other transports [RFC8311]. For instance:

* The sender can test the integrity of a small random sample of the receiver's feedback by occasionally setting the IP-ECN field to a value normally only set by the network. Then, it can test whether the receiver's feedback faithfully reports what it expects (see Paragraph 2 of Section 20.2 of the ECN spec [RFC3168]. This works for loss, and it will work for the accurate ECN feedback [RFC7560] intended for L4S. Like the (Historic) ECN nonce spec, this technique does not protect against a misbehaving sender. But it allows a well-behaved sender to check that each receiver is correctly feeding back congestion notifications.

* A network can check that its ECN markings (or packet losses) have been passed correctly around the full feedback loop by auditing

Congestion Exposure (ConEx) [RFC7713]. This assures that the integrity of congestion notifications and feedback messages must have both been preserved. ConEx information is also available anywhere along the network path, so it can be used to enforce a congestion response. Whether the receiver or a downstream network is suppressing congestion feedback or the sender is unresponsive to the feedback, or both, ConEx is intended to neutralize any advantage that any of these three parties would otherwise gain.

* Congestion feedback fields in transport-layer headers are immutable end to end and therefore amenable to end-to-end integrity protection. This preserves the integrity of a receiver's feedback messages to the sender, but it does not protect against misbehaving receivers or misbehaving senders. The TCP Authentication Option (TCP-AO) [RFC5925], QUIC's end-to-end protection [RFC9001], or end-to-end IPsec integrity protection [RFC4303] can be used to detect any tampering with congestion feedback (whether malicious or accidental), respectively, in TCP, QUIC, or any transport. TCP-AO covers the main TCP header and TCP options by default, but it is often too brittle to use on many end-to-end paths, where middleboxes can make verification fail in their attempts to improve performance or security, e.g., by resegmentation or shifting the sequence space.

At the time of writing, it is becoming common to protect the integrity of transport feedback using QUIC. However, it is still not common to protect the integrity of the wider congestion feedback loop, whether based on loss or Classic ECN. If this position changes during the L4S experiment, one or more of the above techniques might need to be developed and deployed.

C.2. Notification of Less Severe Congestion than CE

Various researchers have proposed to use ECT(1) as a less severe congestion notification than CE, particularly to enable flows to fill available capacity more quickly after an idle period, when another flow departs or when a flow starts, e.g., the Variable-structure congestion Control Protocol (VCP) [VCP] and Queue View (QV) [QV].

Before assigning ECT(1) as an identifier for L4S, we must carefully consider whether it might be better to hold ECT(1) in reserve for future standardization of rapid flow acceleration, which is an important and enduring problem [RFC6077].

Pre-Congestion Notification (PCN) is another scheme that assigns alternative semantics to the IP-ECN field. It uses ECT(1) to signify a less severe level of pre-congestion notification than CE [RFC6660]. However, the IP-ECN field only takes on the PCN semantics if packets carry a Diffserv codepoint defined to indicate PCN marking within a controlled environment. PCN is required to be applied solely to the outer header of a tunnel across the controlled region in order not to interfere with any end-to-end use of the ECN field. Therefore, a PCN region on the path would not interfere with the L4S service identifier defined in Section 2.

Acknowledgements

Thanks to Richard Scheffenegger, John Leslie, David Täht, Jonathan Morton, Gorry Fairhurst, Michael Welzl, Mikael Abrahamsson, and Andrew McGregor for the discussions that led to this specification. Ing-jyh (Inton) Tsang was a contributor to the early draft versions of this document. Thanks to Mikael Abrahamsson, Lloyd Wood, Nicolas Kuhn, Greg White, Tom Henderson, David Black, Gorry Fairhurst, Brian Carpenter, Jake Holland, Rod Grimes, Richard Scheffenegger, Sebastian Moeller, Neal Cardwell, Praveen Balasubramanian, Reza Marandian Hagh, Pete Heist, Stuart Cheshire, Vidhi Goel, Mirja Kühlewind, Ermin Sakic, and Martin Duke for providing help and reviewing this document. And thanks to Ingemar Johansson for reviewing and providing substantial text. Thanks also to the area reviewers: Valery Smyslov, Maria Ines Robles, Bernard Aboba, Lars Eggert, Roman Danyliw, and Éric Vyncke. Thanks to Sebastian Moeller for identifying the interaction with VPN anti-replay and to Jonathan Morton for identifying the attack based on this. Particular thanks to tsvwg chairs Gorry Fairhurst, David Black, and Wes Eddy for patiently helping this and the other L4S documents through the IETF process. Appendix A, which lists the Prague L4S Requirements, is based on text authored by Marcelo Bagnulo Braun that was originally an appendix to [RFC9330]. That text was in turn based on the collective output of the attendees listed in the minutes of a 'bar BoF' on DCTCP Evolution during IETF 94 [TCPPrague].

The authors' contributions were partly funded by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700). The contribution of Koen De Schepper was also partly funded by the 5Growth and DAEMON EU H2020 projects. Bob Briscoe was partly funded by the Research Council of Norway through the TimeIn project, CableLabs, and the Comcast Innovation Fund. The views expressed here are solely those of the authors.

Authors' Addresses

Koen De Schepper
Nokia Bell Labs
Antwerp
Belgium
Email: koen.de_schepper@nokia.com
URI: https://www.bell-labs.com/about/researcher-profiles/
koende_schepper/

Bob Briscoe (editor)
Independent
United Kingdom
Email: ietf@bobbriscoe.net
URI: https://bobbriscoe.net/