Transport Services (tsv)
Internet-Draft
Intended status: Experimental
Expires: May 19, 2021

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

November 15, 2020

Identifying Modified Explicit Congestion Notification (ECN) Semantics for Ultra-Low Queuing Delay (L4S)
draft-ietf-tsvwg-ecn-l4s-id-12

Abstract

This specification defines the identifier to be used on IP packets for a new network service called low latency, low loss and scalable throughput (L4S). L4S uses an Explicit Congestion Notification (ECN) scheme that is similar to the original (or 'Classic') ECN approach. 'Classic' ECN marking was required to be equivalent to a drop, both when applied in the network and when responded to by a transport. Unlike 'Classic' ECN marking, for packets carrying the L4S identifier, the network applies marking more immediately and more aggressively than drop, and 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 non-L4S flow under the same conditions. Nonetheless, the much more frequent control signals and the finer responses to them result in much more fine-grained adjustments, so that ultra-low and consistently low queuing delay (typically sub-millisecond on average) 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. It gives an incremental migration path so that suitably modified network bottlenecks can distinguish and isolate existing traffic that still follows the Classic behaviour, to prevent it degrading the low queuing delay and low loss of L4S traffic. This specification defines the rules that L4S transports and network elements need to follow to ensure they neither harm each other's performance nor that of Classic traffic. Examples of new active queue management (AQM) marking algorithms and examples of new transports (whether TCP-like or real-time) are specified separately.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This specification defines the identifier to be used on IP packets for a new network service called low latency, low loss and scalable throughput (L4S). It is similar to the original (or 'Classic') Explicit Congestion Notification (ECN [RFC3168]). RFC 3168 required 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, the network applies L4S marking more immediately and more aggressively than drop, and 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 non-L4S flow under the same conditions. Nonetheless, the much more frequent control signals and the finer responses to them result in ultra-low queuing delay without compromising link utilization, and this low delay can be maintained during high load. Ultra-low queuing delay means less than 1 millisecond (ms) on average and less than about 2 ms at the 99th percentile.

An example of a scalable congestion control that would enable the L4S service is Data Center TCP (DCTCP), which until now has been applicable solely to controlled environments like data centres [RFC8257], because it is too aggressive to co-exist with existing TCP-Reno-friendly traffic. The DualQ Coupled AQM, which is defined in a complementary experimental specification [I-D.ietf-tsvwg-aqm-dualq-coupled], is an AQM framework that enables scalable congestion controls like DCTCP to co-exist with existing traffic, each getting roughly the same flow rate when they compete under similar conditions. Note that a transport such as DCTCP 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 of the SCReAM [RFC8298] real-time media congestion avoidance technique (RMCAT). The factor that distinguishes L4S from Classic traffic is its behaviour in response to congestion. The transport wire protocol, e.g. TCP, QUIC, SCTP, DCCP, 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. It gives an incremental migration path so that suitably modified network

bottlenecks can distinguish and isolate existing Classic traffic from L4S traffic to prevent it from degrading the ultra-low delay and loss of the new scalable transports, without harming Classic performance. Initial implementation of the separate parts of the system has been motivated by the performance benefits.

1.1. Latency, Loss and Scaling Problems

Latency is becoming the critical performance factor for many (most?) applications on the public Internet, e.g. interactive Web, Web services, voice, conversational video, interactive video, interactive remote presence, instant messaging, online gaming, remote desktop, cloud-based applications, and video-assisted remote control of machinery and industrial processes. In the 'developed' world, further increases in access network bit-rate offer diminishing returns, whereas latency is still a multi-faceted problem. In the last decade or so, much has been done to reduce propagation time by placing caches or servers closer to users. However, queuing remains a major intermittent component of latency.

The Diffserv architecture provides Expedited Forwarding [RFC3246], so that low latency traffic can jump the queue of other traffic. However, on access links dedicated to individual sites (homes, small enterprises or mobile devices), often all traffic at any one time will be latency-sensitive. Then, given nothing to differentiate from, Diffserv makes no difference. Instead, we need to remove the 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. These delays appear only intermittently--only when a capacity-seeking (e.g. TCP) flow is long enough for the queue to fill the buffer, making every packet in other flows sharing the buffer sit through the queue.

Active queue management (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 (aka. greedy) flows to keep the buffer empty for its intended purpose: absorbing bursts. However, RED [RFC2309] and other algorithms from the 1990s were sensitive to their configuration and hard to set correctly. So, this form of AQM was not widely deployed.

More recent state-of-the-art AQM methods, e.g. FQ-CoDel [RFC8290], PIE [RFC8033], Adaptive RED [ARED01], are easier to configure,

because they define the queuing threshold in time not bytes, so it is invariant for different link rates. However, no matter how good the AQM, the sawtoothing sending window of a Classic congestion control will either cause queuing delay to vary or cause the link to be under-utilized. Even with a perfectly tuned AQM, the additional queuing delay will be of the same order as the underlying speed-of-light delay across the network.

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.

Altering host behaviour is not enough on its own though. Even if hosts adopt low latency behaviour (scalable congestion controls), they need to be isolated from the behaviour of existing Classic congestion controls that induce large queue variations. L4S enables that migration by providing latency isolation in the network and distinguishing 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 [I-D.ietf-tsvwg-aqm-dualq-coupled] is sufficient, and actually gives better tail latency. Both approaches are addressed in this document.

The DualQ solution was developed to make ultra-low latency available without requiring per-flow queues at every bottleneck. This was because 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 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 (footnote 6 of Jacobson and Karels [TCP-CA]). Given regular broadband bit-rates over WAN distances are already [RFC3649] beyond the scaling range of Reno TCP, 'less unscalable' Cubic [RFC8312] and Compound [I-D.sridharan-tcpm-ctcp] variants of TCP have been successfully deployed. However, these are now approaching their scaling limits. Unfortunately, fully scalable congestion controls such as DCTCP [RFC8257] cause Classic ECN congestion controls sharing the same queue to starve themselves, which is why they have been confined to private data centres or research testbeds (until now).

It turns out that a congestion control algorithm like DCTCP that solves the latency problem also solves the scalability problem of Classic congestion controls. The finer sawteeth in the congestion window 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 [DCttH15] 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 summarised without the maths in the L4S architecture document [I-D.ietf-tsvwg-l4s-arch].

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 [RFC2119]. In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

Classic Congestion Control: A congestion control behaviour that can co-exist with standard TCP Reno [RFC5681] without causing significantly negative impact on its flow rate [RFC5033]. With Classic congestion controls, as flow rate scales, the number of round trips between congestion signals (losses or ECN marks) rises with the flow rate. So it takes longer and longer to recover after each congestion event. Therefore control of queuing and utilization becomes very slack, and the slightest disturbance prevents a high rate from being attained [RFC3649].

For instance, with 1500 byte packets and an end-to-end round trip time (RTT) of 36 ms, over the years, as Reno flow rate scales from 2 to 100 Mb/s the number of round trips taken to recover from a congestion event rises proportionately, from 4 round trips to 200. Cubic [RFC8312] was developed to be less unscalable, but it is approaching its scaling limit; with the same RTT of 36ms, at 100Mb/s it takes about 106 round trips to recover, and at 800 Mb/s its recovery time triples to over 340 round trips, or still more than 12 seconds (Reno would take 57 seconds). Cubic only becomes significantly better than Reno at high delay and rate combinations, for example at 90 ms RTT and 800 Mb/s a Reno flow takes 4000 RTTs or 6 minutes to recover, whereas Cubic 'only' needs 188 RTTs, which is still 17 seconds (double its recovery time at 100Mb/s).

Scalable Congestion Control: A congestion control where the average time from one congestion signal to the next (the recovery time) remains invariant as the flow rate scales, all other factors being equal. This maintains the same degree of control over queueing 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 [Mathis09], TCP Prague [PragueLinux] and the L4S variant of SCREAM for real-time media [RFC8298]). See Section 4.3 for more explanation.

Classic service: The Classic service is intended for all the congestion control behaviours that co-exist with Reno [RFC5681] (e.g. Reno itself, Cubic [RFC8312], Compound [I-D.sridharan-tcpm-ctcp], TFRC [RFC5348]). The term 'Classic queue' means a queue providing the Classic service.

Low-Latency, Low-Loss Scalable throughput (L4S) service: The 'L4S' service is intended for traffic from scalable congestion control algorithms, such as Data Center TCP [RFC8257]. The L4S service is for more general traffic than just DCTCP--it allows the set of congestion controls with similar scaling properties to DCTCP to evolve, such as the examples listed above (Relentless, Prague, SCReAM). 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', '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, as long as it does not build a queue (e.g. DNS, VoIP, game sync datagrams, etc).

Reno-friendly: The subset of Classic traffic that excludes unresponsive traffic and excludes experimental congestion controls intended to coexist with Reno but without always being strictly friendly to Reno (as allowed by [RFC5033]). Reno-friendly is used in place of 'TCP-friendly', given that the TCP protocol is used with many different congestion control behaviours.

Classic ECN: The original Explicit Congestion Notification (ECN) protocol [RFC3168], which requires ECN signals to be treated the same as drops, both when generated in the network and when responded to by the sender. The names used for the four codepoints of the 2-bit IP-ECN field are as defined in [RFC3168]:

Not ECT, ECT(0), ECT(1) and CE, where ECT stands for ECN-Capable Transport and CE stands for Congestion Experienced.

1.3. Scope

The new L4S identifier defined in this specification is applicable for IPv4 and IPv6 packets (as for 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 intended for experimental status, 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 track 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);
- changes the status of the experimental ECN nonce [RFC3540] to historic;
- makes consequent updates to the following additional proposed standard RFCs to reflect the above two bullets:
  - ECN for RTP [RFC6679];
  - 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. Consensus Choice of L4S Packet Identifier: Requirements

This subsection briefly records the process that led to a consensus choice of L4S identifier, selected from all the alternatives in Appendix B.

The identifier for packets using the Low Latency, Low Loss, Scalable throughput (L4S) service needs to meet the following requirements:

- it SHOULD survive end-to-end between source and destination applications: 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 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;
- 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 those more likely to succeed.

It is recognised that the chosen identifier is unlikely to satisfy all these requirements, particularly given the limited space left in the IP header. Therefore a compromise will be necessary, which is why all the above requirements are expressed with the word 'SHOULD' not 'MUST'. Appendix B discusses the pros and cons of the compromises made in various competing identification schemes against the above requirements.

On the basis of this analysis, "ECT(1) and CE codepoints" is the best compromise. Therefore this scheme is defined in detail in the following sections, while Appendix B records the rationale for this decision.

3. L4S Packet Identification at Run-Time

The L4S treatment is an experimental track alternative packet marking treatment [RFC4774] 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. Like Classic ECN, L4S ECN identifies both network and host behaviour: it identifies the marking treatment that network nodes are expected to apply to L4S packets, and it identifies packets that have been sent from hosts that are expected to comply with a broad type of sending behaviour.

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 normally classifies arriving ECT(1) and CE packets for L4S treatment. 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.

4. Prerequisite Transport Layer Behaviour

4.1. Prerequisite 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 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 [I-D.ietf-tcpm-accurate-ecn]) 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 must be supporting 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.

SCTP: A suitable ECN feedback mechanism for SCTP could add a chunk to report the number of received CE marks (e.g. [I-D.stewart-tsvwg-sctpecn]), and update the ECN feedback protocol sketched out in Appendix A of the 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 [I-D.ietf-avtcore-cc-feedback-message]. 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 the v1 IETF QUIC transport [I-D.ietf-quic-transport].

DCCP: The 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 time from one ECN congestion signal to the next (the 'recovery time') does not increase as flow rate scales, all other factors being equal. This is termed a scalable congestion control. This is necessary to ensure that queue variations remain small as flow rate scales, without having to sacrifice utilization.

For instance, for DCTCP, TCP Prague [PragueLinux] and the L4S variant of SCReAM [RFC8298], the average recovery time is always half a round trip, whatever the flow rate.

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

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, DCTCP's 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 MUST NOT tag its packets with the ECT(1) codepoint unless it complies with the following bulleted requirements. The specification of a particular scalable congestion control MUST describe in detail how it satisfies each requirement and, for any non-mandatory requirements, it MUST justify why it does not comply:

- As well as responding to ECN markings, a scalable congestion control MUST react to packet loss in a way that will coexist safely with a TCP Reno congestion control [RFC5681] (see Appendix A.1.3 for rationale).

- A scalable congestion control MUST implement monitoring in order to detect a likely non-L4S but ECN-capable AQM at the bottleneck. On detection of a likely ECN-capable bottleneck it SHOULD be capable (dependent on configuration) of automatically adapting its congestion response to coexist with TCP Reno congestion controls [RFC5681] (see Appendix A.1.4 for rationale and a referenced algorithm).

  Note that a scalable congestion control is not expected to change to setting ECT(0) while it falls back to coexist with Reno.

- A scalable congestion control MUST eliminate RTT bias as much as possible in the range between the minimum likely RTT and typical RTTs expected in the intended deployment scenario (see Appendix A.1.5 for rationale).

- 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 (see Appendix A.1.6 for rationale).

- A scalable congestion control SHOULD detect loss by counting in time-based units, which is scalable, as opposed to counting in units of packets (as in the 3 DupACK rule of RFC 5681 TCP), which is not scalable. As packet 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.1.7 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.

- 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 under the worst-case assumption that the bottleneck link capacity equals the current flow rate. 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. Instead, it is only required that the specification of a particular scalable congestion control MUST define, quantify and justify its approach to limiting bursts.

To participate in the L4S experiment, a scalable congestion control MUST be capable of being replaced by a Classic congestion control (by application and by administrative control). A purely Classic congestion control will not tag its packets with the ECT(1) codepoint.

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 (Section 5.4.1.1) this document discusses the conditions for mixing other "'Safe' Unresponsive Traffic" (e.g. DNS, LDAP, NTP, voice, game sync packets) with L4S traffic. 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.

4.4. Filtering or Smoothing of ECN Feedback

Section 5.2 below specifies that an L4S AQM is expected to signal L4S ECN without 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 [I-D.ietf-tsvwg-l4s-arch], 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 all the flows, so it has to smooth out variations for a worst-case

RTT to ensure stability. For all the typical flows with shorter RTT 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. Nonetheless, the specification of a particular L4S congestion control SHOULD describe how it smooths the L4S ECN signals fed back to it from the receiver.

5. Prerequisite Network Node Behaviour

5.1. Prerequisite Classification and Re-Marking Behaviour

A network node that implements the L4S service MUST classify arriving ECT(1) packets for L4S treatment and, other than in the exceptional case referred to next, it MUST classify arriving CE packets for L4S treatment as well. 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.1 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 RFC 3168. Specifically, the ECT(1) codepoint MUST NOT be changed to any other codepoint 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 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 using Classic drop until the overload episode has subsided, as recommended for all AQM methods in [RFC7567] (Section 4.2.1), which follows the similar advice in RFC 3168 (Section 7). Where an L4S AQM is transport-aware, this requirement could be satisfied by using Classic drop in only the most overloaded individual per-flow AQMs or in a DualQ by redirecting packets in those flows contributing most to the overload from the L4S queue so that they are subjected to drop in the Classic queue [I-D.briscoe-docsis-q-protection].

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, it will do so under the same conditions as it would drop Not-ECT packets [RFC3168]. 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 [I-D.ietf-tsvwg-aqm-dualq-coupled]).

5.2. The Meaning of L4S CE Relative to Drop

The likelihood that an 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

$$p_C \approx (p_L / k)^2$$

The constant of proportionality (k) does not have to be standardised 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.

The relative strengths of L4S CE and drop are irrelevant in an AQM that schedules application flows explicitly (e.g. an FQ scheduler). Nonetheless, the above relationship defines the coupling between L4S and Classic congestion signals in a DualQ Coupled AQM [I-D.ietf-tsvwg-aqm-dualq-coupled].

Note that, contrary to RFC 3168, a Dual Queue Coupled 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 RFC 3168. However, if it marks ECT(0) packets, it does so under the same conditions that it would have dropped a Not-ECT packet.

Also, L4S CE marking needs to be interpreted as an unsmoothed signal, in contrast to the Classic approach in which AQMs filter out variations before signalling congestion. An L4S AQM SHOULD NOT smooth or filter out variations in the queue before signalling congestion. In the L4S architecture [I-D.ietf-tsvwg-l4s-arch], the sender, not the network, is responsible for smoothing out variations.

This requirement is worded as 'SHOULD NOT' rather than 'MUST NOT' to allow for the case where the signals from a Classic smoothed AQM are coupled with those from an unsmoothed L4S AQM. Nonetheless, the spirit of the requirement is for all systems to expect that L4S ECN signalling is unsmoothed and unfiltered, which is important for interoperability.

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

To implement the L4S treatment, a network node does not need to identify transport-layer flows. Nonetheless, if an implementer is willing to identify transport-layer flows at a network node, and if the most recent ECT packet in the same flow was ECT(0), the node MAY classify CE packets for Classic ECN [RFC3168] treatment. In all other cases, a network node MUST classify all CE packets for L4S treatment. 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 the most recent ECT packet in a flow was ECT(1).

If an implementer uses flow-awareness to classify CE packets, to determine whether the flow is using ECT(0) or ECT(1) it only uses the most recent ECT packet of a flow (this advice will need to be verified as part of L4S experiments). This is because a sender might switch from sending ECT(1) (L4S) packets to sending ECT(0) (Classic ECN) packets, or back again, in the middle of a transport-layer flow (e.g. it might manually switch its congestion control module mid-connection, or it might be deliberately attempting to confuse the network).

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 Coupled DualQ AQM [I-D.ietf-tsvwg-aqm-dualq-coupled], 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, game sync packets) into the low latency queue to mix with L4S traffic. Such non-ECN-based packet types MUST be safe to mix with L4S traffic without harming the low latency service, where 'safe' is explained in Section 5.4.1.1.1 below.

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;

- 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. Therefore, a network element that implements L4S in a shared queue MAY classify additional packets into the L queue if they carry certain non-ECN identifiers. For instance:

- addresses of specific applications or hosts configured to be safe (or perhaps they comply with L4S behaviour and can respond to ECN feedback, but perhaps cannot set the ECN field for some reason);

- certain protocols that are usually lightweight (e.g. ARP, DNS);

- specific Diffserv codepoints that indicate traffic with limited burstiness such as the EF (Expedited Forwarding [RFC3246]), Voice-Admit [RFC5865] or proposed NQB (Non-Queue-Building [I-D.ietf-tsvwg-nqb]) service classes or equivalent local-use DSCPs (see [I-D.briscoe-tsvwg-l4s-diffserv]).

Of course, a packet that carried both the ECT(1) codepoint and a non-ECN identifier associated with the L queue would be classified into the L queue.

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.

To include additional traffic with L4S, a network element only reads identifiers such as those itemized above. It MUST NOT alter these non-ECN identifiers, so that they survive for any potential use later on the network path.

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 completely regularly) 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, keep-alives, could match this description.

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. initially the operator may wish to confine the benefits of L4S to business customers).

In this exclusion case, the operator 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.

The operator MUST NOT alter the end-to-end L4S ECN identifier from L4S to Classic, because its 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.

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. The two can be combined if a network operator wants to control bandwidth allocation but it also wants to provide low latency - for any amount of traffic within one of these allocations of bandwidth (rather than only providing low latency by limiting bandwidth) [I-D.briscoe-tsvwg-l4s-diffserv].

The DualQ examples so far have been framed in the context of providing the default Best Efforts 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 with a low latency and a Classic variant.

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 (say) 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).
[I-D.briscoe-tsvwg-l4s-diffserv] gives a number of examples of such arrangements to address various requirements.

[I-D.briscoe-tsvwg-l4s-diffserv] describes how an operator might use L4S to offer low latency for all L4S traffic as well as using 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 'Recommended-standard-use' could be set by the sending host or a network. Those tagged '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 queueing (e.g. FQ-CoDel [RFC8290]), the L4S identifier could complement the Layer-4 flow ID as a further level of flow granularity (i.e. Not-ECT and ECT(0) queued separately from ECT(1) and CE packets). "Risk of reordering Classic CE packets" in Appendix B.1 discusses the resulting ambiguity if packets originally marked 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 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 Supporting L4S AQMs

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 tradeoff, because batch-handling of packets is done for good reason, e.g. 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, 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).

6. 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 WG 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-D.ietf-tsvwg-aqm-dualq-coupled] sets operational and management requirements for experiments with DualQ Coupled AQMs; and General operational and management requirements for experiments with L4S congestion controls are given in Section 4 and Section 5 above, e.g. co-existence and scaling requirements, 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.

6.1. 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?

- Does use of L4S over the Internet result in 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
  - scaling range of flow rates and RTTs
- How much does burstiness in the Internet affect L4S performance, and how much limitation of bustiness was needed and/or was realized - both at senders and at links, especially radio links?
- 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.

6.2. 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?

- Fixing the poor Interaction between current L4S congestion controls and CoDel with only Classic ECN support during flow startup

6.3. 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 bocks (e.g. AccECN, 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

- Novel applications enabled by L4S

7. IANA Considerations

The 01 codepoint of the ECN Field of the IP header is specified by the present 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].

When the present document is published as an RFC, IANA is asked to update the 01 entry in the registry, "ECN Field (Bits 6-7)" to the

following (see https://www.iana.org/assignments/dscp-registry/dscp-registry.xhtml#ecn-field):

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

[XXXX is the number that the RFC Editor assigns to the present document (this sentence to be removed by the RFC Editor)].

8. 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 [I-D.ietf-tsvwg-l4s-arch] for further discussion of mis-use of the identifier, as well as extensive discussion of policing rate and latency in regard to L4S.

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

9. Acknowledgements

Thanks to Richard Scheffenegger, John Leslie, David Taeht, 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 drafts of this document. And thanks to Mikael Abrahamsson, Lloyd Wood, Nicolas Kuhn, Greg White, Tom Henderson, David Black, Gorry Fairhurst, Brian Carpenter, Jake Holland, Rod Grimes and Richard Scheffenegger for providing help and reviewing this draft and to Ingemar Johansson for reviewing and providing substantial text. Particular thanks to Wes Eddy for patiently shepherding this and the other L4S drafts through the IETF process. Appendix A listing the Prague L4S Requirements is based on text authored by Marcelo Bagnulo Braun that was originally an appendix to [I-D.ietf-tsvwg-l4s-arch]. 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 part-funded by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700). Bob Briscoe was also funded partly by the Research Council of Norway through the TimeIn

project, partly by CableLabs and partly by the Comcast Innovation Fund. The views expressed here are solely those of the authors.

10. References

10.1. Normative References

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

[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 64(6):1522-1533, June 2015, <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6871352>.

[Ahmed19] Ahmed, A., "Extending TCP for Low Round Trip Delay", Masters Thesis, Uni 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", ACM SIGMETRICS 2011 , June 2011.

[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 , August 2001, <http://www.icir.org/floyd/red.html>.

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

[ecn-fallback] Briscoe, B. and A. Ahmed, "TCP Prague Fall-back on Detection of a Classic ECN AQM", bobbriscoe.net Technical Report TR-BB-2019-002, April 2020, <https://arxiv.org/abs/1911.00710>.

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

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

[I-D.ietf-avtcore-cc-feedback-message] Sarker, Z., Perkins, C., Singh, V., and M. Ramalho, "RTP Control Protocol (RTCP) Feedback for Congestion Control", draft-ietf-avtcore-cc-feedback-message-09 (work in progress), November 2020.

[I-D.ietf-quic-transport] Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-32 (work in progress), October 2020.

[I-D.ietf-tcpm-accurate-ecn] Briscoe, B., Kuehlewind, M., and R. Scheffenegger, "More Accurate ECN Feedback in TCP", draft-ietf-tcpm-accurate-ecn-13 (work in progress), November 2020.

[I-D.ietf-tcpm-generalized-ecn] Bagnulo, M. and B. Briscoe, "ECN++: Adding Explicit Congestion Notification (ECN) to TCP Control Packets", draft-ietf-tcpm-generalized-ecn-06 (work in progress), October 2020.

[I-D.ietf-tcpm-rack] Cheng, Y., Cardwell, N., Dukkipati, N., and P. Jha, "The RACK-TLP loss detection algorithm for TCP", draft-ietf-tcpm-rack-13 (work in progress), November 2020.

[I-D.ietf-tsvwg-aqm-dualq-coupled]
Schepper, K., Briscoe, B., and G. White, "DualQ Coupled AQMs for Low Latency, Low Loss and Scalable Throughput (L4S)", draft-ietf-tsvwg-aqm-dualq-coupled-12 (work in progress), July 2020.

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

[I-D.ietf-tsvwg-l4s-arch]
Briscoe, B., Schepper, K., Bagnulo, M., and G. White, "Low Latency, Low Loss, Scalable Throughput (L4S) Internet Service: Architecture", draft-ietf-tsvwg-l4s-arch-07 (work in progress), October 2020.

[I-D.ietf-tsvwg-nqb]
White, G. and T. Fossati, "A Non-Queue-Building Per-Hop Behavior (NQB PHB) for Differentiated Services", draft-ietf-tsvwg-nqb-03 (work in progress), November 2020.

[I-D.ietf-tsvwg-rfc6040update-shim]
Briscoe, B., "Propagating Explicit Congestion Notification Across IP Tunnel Headers Separated by a Shim", draft-ietf-tsvwg-rfc6040update-shim-10 (work in progress), March 2020.

[I-D.morton-tsvwg-sce]
Morton, J., Heist, P., and R. Grimes, "The Some Congestion Experienced ECN Codepoint", draft-morton-tsvwg-sce-02 (work in progress), November 2020.

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

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

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

[Mathis09]
Mathis, M., "Relentless Congestion Control", PFLDNeT'09 , May 2009, <http://www.hpcc.jp/pfldnet2009/Program_files/1569198525.pdf>.

[Paced-Chirping]
Misund, J., "Rapid Acceleration in TCP Prague", Masters Thesis , May 2018, <https://riteproject.files.wordpress.com/2018/07/misundjoakimmastersthesissubmitted180515.pdf>.

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

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

[QV] Briscoe, B. and P. Hurtig, "Up to Speed with Queue View", RITE Technical Report D2.3; Appendix C.2, August 2015, <https://riteproject.files.wordpress.com/2015/12/rite-deliverable-2-3.pdf>.

[RFC2309] Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J., and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, DOI 10.17487/RFC2309, April 1998, <https://www.rfc-editor.org/info/rfc2309>.

[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>.

[RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, DOI 10.17487/RFC2983, October 2000, <https://www.rfc-editor.org/info/rfc2983>.

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

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

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

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

[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>.

[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>.

[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>.

[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>.

[Savage-TCP] Savage, S., Cardwell, N., Wetherall, D., and T. Anderson, "TCP Congestion Control with a Misbehaving Receiver", ACM SIGCOMM Computer Communication Review 29(5):71--78, October 1999.

[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, May 2015, <https://arxiv.org/abs/1904.07598>.

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

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

[VCP] Xia, Y., Subramanian, L., Stoica, I., and S. Kalyanaraman, "One more bit is enough", Proc. SIGCOMM'05, ACM CCR 35(4)37--48, 2005, <http://doi.acm.org/10.1145/1080091.1080098>.

Appendix A. 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 necessary safety improvements (requirements) this appendix also includes preferable performance improvements (optimizations).

These recommendations have become know 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, both from 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 it is likely that most of these requirements equally apply to other scalable congestion controls, covering adaptive real-time media, etc., not just capacity-seeking behaviours.

A.1. 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 marked with ECN 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 [I-D.ietf-tcpm-accurate-ecn] specifies an experimental change to the TCP wire protocol to satisfy these requirements. Most other transport protocols already satisfy this requirement (see Section 4.2).

A.1.3. Fall back to Reno-friendly 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 TCP Reno congestion control [RFC5681] (see the precise normative requirement wording in Section 4.3).

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 with this requirement 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 part of the congestion control specification of the relevant transport. In the particular case of DCTCP, the DCTCP specification [RFC8257] states that "It is RECOMMENDED that an implementation deal with loss episodes in the same way as conventional TCP." For safe deployment of a scalable congestion control in the public Internet, the above requirement would need to be defined as a "MUST".

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 bit set and also experience loss. Current DCTCP implementations each react differently to this situation. At least one implementation reacts only to the drop signal (e.g. by halving the CWND) and at least another DCTCP implementation reacts to both signals (e.g. by halving the CWND due to the drop and also further reducing the CWND based on the proportion of marked packet). A third approach for the public Internet has been proposed that adjusts the loss response to result in a halving when combined with the ECN response. We believe that further experimentation is needed to understand what is the best behaviour for the public Internet, which may or not be one of these existing approaches.

A.1.4. Fall back to Reno-friendly congestion control on classic ECN bottlenecks

Description: A scalable congestion control needs to react to ECN marking from a non-L4S, but ECN-capable, bottleneck in a way that will coexist with a TCP Reno congestion control [RFC5681] (see the precise normative requirement wording in Section 4.3).

Motivation: Similarly to the requirement in Appendix A.1.3, this requirement is a safety condition to ensure a scalable congestion control behaves properly when it builds a queue at a network bottleneck that has not been upgraded to support L4S. On detecting Classic ECN marking (see below), a scalable congestion control will need to fall back to Classic congestion control behaviour. If it does not comply with this requirement it could starve Classic traffic.

A passive monitoring algorithm to detect a Classic ECN AQM at the bottleneck is provided in [ecn-fallback], which also provides a link to Linux source code. 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 relevant normative requirement (Section 4.3) is expressed as a 'SHOULD' to allow the possibility that the operator of the host knows that the network it serves has not deployed any single queue classic ECN AQM (e.g. a CDN might be testing out of band for signs of Classic ECN AQMs, or they might have manually checked which ISPs they serve have not deployed Classic ECN AQMs).

Nonetheless, monitoring is still expressed as a 'MUST' because there is still a possibility that there is a Classic ECN AQM somewhere else on the path (to continue the CDN example, perhaps beyond the ISP in a home network). Then, if the server operators have disabled fall-back for parts of their deployment, they can reconsider their policy or at least do more focused testing if in-band monitoring frequently detects single-queue Classic ECN AQMs.

A.1.5. Reduce RTT dependence

Description: A scalable congestion control needs to reduce or eliminate RTT bias 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).

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 1ms and 100ms. If a Classic congestion control induces a 100ms queue, it turns these RTTs into 101ms and 200ms leading to a throughput ratio of about 2:1. Whereas if a scalable congestion control induces only a 1ms 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 this requirement.

The RTT bias in current Classic congestion controls works satisfactorily when the RTT is higher than typical, and L4S does not change that. So, there is no additional requirement for high RTT L4S flows to remove RTT bias - they can but they don't have to.

A.1.6. 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).

Motivation: As currently specified, the minimum required congestion window of TCP (and its derivatives) is set to 2 sender maximum segment sizes (SMSS) (see equation (4) in [RFC5681]). Once the congestion window reaches this minimum, all known window-based congestion control algorithms become unresponsive to congestion signals. No matter how much drop or ECN marking, the congestion window of all these algorithms 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.

L4S mechanisms significantly reduce queueing 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 6ms or less, which is quite common when accessing a nearby data centre. So, any more than two such parallel TCP flows will become unresponsive and increase queuing delay.

Unless scalable congestion controls address this requirement from the start, they will frequently become unresponsive, 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 2 SMSS. For instance, one possible mechanism that can maintain a congestion window significantly less than 1 SMSS is described in [Ahmed19], and other approaches are likely to be feasible.

However, the requirement in Section 4.3 is worded as a "SHOULD" because 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 this requirement as a "SHOULD", specifications of scalable congestion controllers will be able to choose an appropriate minimum window, but they will at least have to justify the decision.

A.1.7. Measuring Reordering Tolerance in Time Units

Description: A scalable congestion control needs to detect loss by counting in time-based units, which is scalable, rather than counting in units of packets, which is not (see the precise normative requirement wording in Section 4.3).

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 [I-D.ietf-tcpm-rack]).

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, covered later). Therefore requiring L4S hosts to detect loss in time-based units would not be a burden.

If this requirement is not placed on L4S hosts, even though it would be no burden on them to do so, all networks will face unnecessary uncertainty over whether some L4S hosts might be detecting loss by counting packets. Then _all_ link technologies will 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 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 speed of loss recovery 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 TCP RACK (Recent ACKnowledgements) [I-D.ietf-tcpm-rack]. 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 RoCEv2 for RDMA). 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. It is believed that such implementations 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 TCP Control Packets and Retransmissions

Description: This item only concerns TCP and its derivatives (e.g. SCTP), because the original specification of ECN for TCP precluded the use of ECN on control packets and retransmissions. 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 derivatives of TCP, e.g. SCTP.

Motivation: RFC 3168 prohibits the use of ECN on these types of TCP packet, 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. [I-D.ietf-tcpm-generalized-ecn] counters each argument in RFC 3168 in turn, showing it was over-cautious. Instead it proposes experimental use of ECN on all types of TCP packet as long as AccECN feedback [I-D.ietf-tcpm-accurate-ecn] is available (which is itself a prerequisite for using a scalable congestion control).

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 traditional TCP Reno additive increase in 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 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 1.2. Even when out of its Reno-compatibility mode, every 8x scaling of Cubic's flow rate leads to 2x 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 (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: Particularly when a flow starts, scalable congestion controls need to converge (reach their steady-state share of the capacity) at least as fast as Classic congestion controls and preferably faster. 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 a factor of 1.5 to 2 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 [Paced-Chirping] 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. Alternative Identifiers

This appendix is informative, not normative. It records the pros and cons of various alternative ways to identify L4S packets to record the rationale for the choice of ECT(1) (Appendix B.1) as the L4S identifier. At the end, Appendix B.8 summarises the distinguishing features of the leading alternatives. It is intended to supplement, not replace the detailed text.

The leading solutions all use the ECN field, sometimes in combination with the Diffserv field. This is because L4S traffic has to indicate that it is ECN-capable anyway, because ECN is intrinsic to how L4S works. Both the ECN and Diffserv fields have the additional advantage that they are no different in either IPv4 or IPv6. A couple of alternatives that use other fields are mentioned at the end, but it is quickly explained why they are not serious contenders.

B.1. ECT(1) and CE codepoints

Definition:

Packets with ECT(1) and conditionally packets with CE would signify L4S semantics as an alternative to the semantics of Classic ECN [RFC3168], specifically:

* The ECT(1) codepoint would signify that the packet was sent by an L4S-capable sender.

* Given shortage of codepoints, both L4S and Classic ECN sides of an AQM would have to use the same CE codepoint to indicate that a packet had 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 would be 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 marked ECT(0). However, the L4S service ought not to need tranport-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 ECN in an AQM acting in a buffer below the IP layer [I-D.ietf-tsvwg-ecn-encap-guidelines]. In such cases, the L4S service would have to drop rather than mark frames even though they might encapsulate an ECN-capable packet.

Risk of reordering Classic CE packets: 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 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 while the second supported L4S ECN marking, which would be the only scenario where some ECT(0) packets could be CE marked by an AQM supporting Classic ECN then the remainder experienced 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,... and 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 [I-D.ietf-tcpm-rack] 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 due to CE being ambiguous will generally be vanishingly small.

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.1.4). Otherwise, if long-running Classic flow(s) are sharing a Classic ECN AQM bottleneck with long-running L4S flow(s), which then apply an L4S response to Classic CE signals, the L4S flows would outcompete the Classic flow(s). 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 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 [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-CoDel 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 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.1.4 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, its 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: The Classic ECN RFCs [RFC3168] and [RFC5562] require a sender to clear the 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 ECN field, these 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 control packets more vulnerable to loss and delay. To address this problem, [I-D.ietf-tcpm-generalized-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 e2e: The ECN field generally works end-to-end across the Internet. Unlike the DSCP, the setting of the ECN field is at least forwarded unchanged by networks that do not support ECN, and networks rarely clear it to zero.

Should work in tunnels: Unlike Diffserv, ECN is defined to always work across tunnels. This scheme works within a tunnel that propagates the ECN field in any of the variant ways it has been defined, from the year 2001 [RFC3168] onwards. However, it is likely that some tunnels still do not implement ECN propagation at all.

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 had reduced to zero, or near-zero, which might never happen.

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

B.2. ECN-DualQ-SCE1

Definition:

In this proposal, an L4S AQM would indicate congestion with ECT(1) in contrast to a Classic AQM, which indicates congestion with CE. More specifically:

* Given shortage of codepoints, with this proposal L4S ECN hosts send packets as ECT(0), like Classic ECN does by default [RFC8311] hosts.

* If the ECT(1) codepoint were used to indicate congestion in this way, it would signify a shallow queue AQM to the end-to-end transport. So those who proposed this approach called it 'Some Congestion Experienced' (SCE) because of its similarity to [I-D.morton-tsvwg-sce]. It has also been described as 'ECT(1) on output', in contrast to the 'ECT(1) on input' approach outlined in Appendix B.1.

* The approach works best if the network is transport-aware and isolates each application flow in its own queue (per-flow queuing, or FQ). Two AQMs are implemented in each queue, one with a shallow target that marks selected ECT packets as ECT(1), the other with a deeper target that marks selected ECT packets as CE, or drops selected non-ECT packets.

* A Classic congestion control would not have the logic to recognize ECT(1) as a congestion signal. So it would (correctly) drive the queue to the deeper threshold, responding only to CE markings. An L4S congestion control that understands this scheme would respond to ECT(1) markings, which ought to therefore keep the queue close to the shallower threshold.

* A dual queue approach has been informally proposed, with an L4S and a Classic queue and coupling similar to [I-D.ietf-tsvwg-aqm-dualq-coupled]. In an interim classification, all ECT packets would be classified into the low latency queue, and non-ECT packets into the Classic queue. But then, in front of the low latency queue, a stateful flow characterization function would maintain a queue occupancy metric. It would then redirect any high occupancy flows into the Classic queue.

Cons:

Network requires transport-layer awareness: There is no variant of this approach that works without network visibility of transport layer flow identifiers (the 5-tuple). Obviously the FQ variant needs to see 5-tuples, but so does the DualQ SCE1 variant (to redirect flows based on sparseness). So there is no arrangement of this approach that operators could choose if they could not access the transport layer, or did not want to (e.g. to support full end-to-end encryption above the IP layer).

Incomplete isolation: When evaluated, the DualQ variant of ECN-DualQ-SCE1 introduced impairments to both L4S and Classic flows. The evaluation used the DOCSIS queue protection function [I-D.briscoe-docsis-q-protection] to maintain the per-flow sparseness metrics and redirect packets from non-sparse flows into the Classic queue. Unfortunately, it is impossible to determine non-sparseness until sufficient packets of each flow have been analyzed. Up to this point, all packets default to the L4S queue. Then:

* Long-running Classic flows experience reordering during the transition to classifying them as Classic. Worse, the reordering occurs early in the flow when it is less robust to confusing RTT measurements;

* Considerable numbers of Classic packets add to the L4S queue - from all the short flows and the start of long flows before the classifier can be certain enough to redirect them to the other (Classic) queue. So true L4S flows unavoidably experience a degree of extra delay.

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

Only recently updated tunnels: If this scheme is applied to an outer header within a tunnel or lower layer encapsulation, the ECT(1) codepoint will be black-holed at decapsulation, unless the decapsulator complies with changes to IP-in-IP tunnels introduced in 2010 [RFC6040], or changes to other tunnels that are (currently) work in progress [I-D.ietf-tsvwg-rfc6040update-shim], [I-D.ietf-tsvwg-ecn-encap-guidelines].

Limited TCP support for feedback: This approach requires transport layer feedback of two congestion signals ECT(1) and CE. Recently developed protocols such as QUIC provide this by default. However, there is limited space in the main TCP header to feed

back both signals reliably and accurately [RFC7560]. AccECN [I-D.ietf-tcpm-accurate-ecn] devotes the limited space in the main TCP header to CE feedback, and optionally feeds back ECT(1) in a new TCP option, which will have limited initial deployment support.

Alters non-participating packets: An AQM following this approach alters some selected ECT(0) packets to ECT(1) irrespective of whether they are participating in the L4S experiment. Although ECT(0) and ECT(1) have historically been defined as equivalent, in practice ECT(1) packets have been extremely rare on the Internet. Therefore, in practice, there might be a risk that firewalls and other devices will block ECT(1) packets, or at least treat them with greater suspicion.

ECN hard in some lower layers: Similarly to the 'Con' point in Appendix B.1, it is not always possible to support ECN in an AQM acting in a buffer below the IP layer [I-D.ietf-tsvwg-ecn-encap-guidelines]. However, adding support to lower layers would be even harder with this scheme, because it needs space for two severity levels of congestion, not one. Without lower layer ECN support, the L4S service would have to drop rather than mark frames even though they might encapsulate an ECN-capable packet. .

Non-L4S service for control packets: Identical to 'Con' point in Appendix B.1.

Pros:

Distinct indication of Classic ECN AQM: An AQM following the ECN-DualQ-SCE1 approach outputs distinctive signals (ECT(1)) compared to those output by a Classic ECN AQM. So an L4S congestion control using the SCE1 approach would inherently respond appropriately to a Classic AQM.

Should work e2e: Identical to 'Pro' point in Appendix B.1.

B.3. ECN-DualQ-SCE0

Definition:

This proposal is the inverse of the ECN-DualQ-SCE1 scheme (see Appendix B.2 above). L4S AQMs signal congestion with the transition ECT(1) -> ECT(0). More specifically:

* L4S senders would send their packets as ECT(1), while Classic ECN senders would continue to send ECT(0) by default [RFC8311].

* FQ AQMs would work in a similar way to that described for ECN-DualQ-SCE1 in Appendix B.2 above. Except the shallow queue AQM would mark selected ECT packets with ECT(0), rather than ECT(1).

  It would seem possible to classify packets by both 5-tuple and ECT codepoint, so that each per-flow queue could instantiate just the one AQM appropriate to the ECT codepoint using it. In this case, CE and Not-ECT packets would be classified into the same queue as ECT(0). However, this would open up the risk of reordering explained below, so it is not considered further.

* A Classic congestion control would only receive CE feedback, and it would have no logic to recognize ECT(0) as congestion markings, because it would send all its packets as ECT(0) anyway. So it would (correctly) drive the queue to the deeper threshold, responding only to CE markings. An L4S congestion control would understand ECT(0) markings as L4S congestion signals and therefore ought to keep the queue close to the shallower threshold.

* Under the SCE0 scheme, a dual queue coupled AQM [I-D.ietf-tsvwg-aqm-dualq-coupled] would use ECT(1) as the L4S classifier in a very similar way to the 'ECT(1) and CE' scheme it was originally designed for. The one difference would be to classify CE packets into the Classic queue along with ECT(0) and Not-ECT.

Cons:

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

Incompatible with all ECN tunnels: The transition ECT(1) -> ECT(0) has never previously been recognized as valid. So, any ECT(0) marking applied to an ECT(1) outer header within a tunnel or lower layer encapsulation will be black-holed at decapsulation by any decapsulator whatever variant of ECN tunnel RFC it complies with.

Limited TCP support for feedback: Identical to 'Con' point in Appendix B.2 above except space would be needed for CE and ECT(0) rather than CE and ECT(1) feedback.

Risk of reordering Classic CE packets: If an L4S flow traverses a path with two or more bottleneck AQMs that both support L4S,

reordering is likely to occur. This is because the first bottleneck will re-mark some ECT(1) packets to ECT(0), which will then be classified into the Classic queue of the second AQM, even though they originated as L4S packets.

In contrast to the 'ECT(1) and CE' scheme in Appendix B.1, the risk of impairment in the ECN-DualQ-SCE0 case is not vanishingly small:

1. Certainly, queuing at more than one bottleneck on the same path would still be quite unusual.

2. However, the ECN-DualQ-SCE0 case occurs if both bottlenecks support L4S ECN and the traffic is L4S. This contrasts with the "ECT(1) and CE" case, which solely occurs if the AQMs are in a certain order (Classic followed by L4S).

3. When misclassification occurs, it is from L4S to Classic. So selected packets are delivered late, which in itself adds delay, and also increases the risk that each late delivery will be deemed a loss and cause a high level of spurious retransmissions. This contrasts with the "ECT(1) and CE" case where selected packets are delivered early, which is very unlikely to have any effect (as already explained in Appendix B.1).

ECN hard in some lower layers: Identical to 'Con' point in Appendix B.2.

Non-L4S service for control packets: Identical to 'Con' point in Appendix B.1.

Pros:

Distinct indication of Classic ECN AQM: An AQM following the ECN-DualQ-SCE0 approach outputs distinctive signals (ECT(0)) compared to those output by a Classic ECN AQM (CE). So an L4S congestion control can inherently respond appropriately to a Classic AQM.

Should work e2e: Identical to 'Pro' point in Appendix B.1.

B.4. ECN Plus a Diffserv Codepoint (DSCP)

Definition:

For packets with a defined DSCP, all codepoints of the ECN field (except Not-ECT) would signify alternative L4S semantics to those for Classic ECN [RFC3168], specifically:

* The L4S DSCP would signify that the packet came from an L4S-capable sender.

* ECT(0) and ECT(1) would both signify that the packet was travelling between transport endpoints that were both ECN-capable.

* CE would signify that the packet had been marked by an AQM implementing the L4S service.

Use of a DSCP is the only approach for alternative ECN semantics given as an example in [RFC4774]. However, it was perhaps considered more for controlled environments than new end-to-end services.

Cons:

Consumes DSCP pairs: A DSCP is by definition not orthogonal to Diffserv. Therefore, wherever the L4S service is applied to multiple Diffserv scheduling behaviours, it would be necessary to replace each DSCP with a pair of DSCPs.

Uses critical lower-layer header space: The resulting increased number of DSCPs might be hard to support for some lower layer technologies, e.g. 802.1Q and MPLS both offer only 3-bits for a maximum of 8 traffic class identifiers. Although L4S should reduce and possibly remove the need for some DSCPs intended for differentiated queuing delay, it will not remove the need for Diffserv entirely, because Diffserv is also used to allocate bandwidth, e.g. by prioritising some classes of traffic over others when traffic exceeds available capacity.

Not end-to-end (host-network): Very few networks honour a DSCP set by a host. Typically a network will zero (bleach) the Diffserv field from all hosts. DSCP bleaching would turn an L4S ECN packet into a Classic ECN packet.

Not end-to-end (network-network): Very few networks honour a DSCP received from a neighbouring network. Typically a network will zero (bleach) the Diffserv field from all neighbouring networks at an interconnection point. Sometimes bilateral arrangements are made between networks, such that the receiving network remarks some DSCPs to those it uses for roughly equivalent services. The likelihood that a DSCP will be bleached or ignored depends on the type of DSCP:

Local-use DSCP: These tend to be used to implement application-specific network policies, but a bilateral arrangement to remark certain DSCPs is often applied to DSCPs in the local-use

range simply because it is easier not to change all of a network's internal configurations when a new arrangement is made with a neighbour.

Recommended standard DSCP: These do not tend to be honoured across network interconnections more than local-use DSCPs. However, if two networks decide to honour certain of each other's DSCPs, the reconfiguration is a little easier if both of their globally recognised services are already represented by the relevant recommended standard DSCPs.

Note that today a recommended standard DSCP gives little more assurance of end-to-end service than a local-use DSCP. In future the range recommended as standard might give more assurance of end-to-end service than local-use, but it is unlikely that either assurance will be high, particularly given the hosts are included in the end-to-end path.

Whenever DSCP bleaching did occur, it would turn an L4S ECN packet into a Classic ECN packet.

Not all tunnels: Diffserv codepoints are often not propagated to the outer header when a packet is encapsulated by a tunnel header. DSCPs are propagated to the outer of uniform mode tunnels, but not pipe mode [RFC2983], and pipe mode is fairly common. Whenever pipe mode was used, it would temporarily turn an L4S ECN packet into a Classic ECN packet.

ECN hard in some lower layers:: Because this approach uses both the Diffserv and ECN fields, an AQM will only work at a lower layer if both can be supported. If individual network operators wished to deploy an AQM at a lower layer, they would usually propagate an IP Diffserv codepoint to the lower layer, using for example IEEE 802.1p. However, the ECN capability is harder to propagate down to lower layers because few lower layers support it.

Hard to distinguish Classic ECN AQM: Defining a DSCP to indicate L4S is a way to help network nodes identify L4S packets (albeit unreliable due to the likelihood of bleaching - see above). However, it does not help hosts distinguish between ECN markings from L4S and Classic AQMs. This is because Classic AQMs would have been implemented without any logic to recognize an L4S DSCP or apply L4S marking behaviour.

Pros:

Could migrate to e2e: If all usage of Classic ECN migrates to usage of L4S, the DSCP would become redundant, and the ECN capability

alone could eventually identify L4S packets without the interconnection problems of Diffserv detailed above, and without having permanently consumed more than one codepoint in the IP header. Although the DSCP does not generally function as an end-to-end identifier (see above), it could be used initially by individual ISPs to introduce the L4S service for their own locally generated traffic.

B.5. ECN capability alone

This approach uses ECN capability alone as the L4S identifier. It would only have been feasible if RFC 3168 ECN had not been widely deployed. This was the case when the choice of L4S identifier was being made and this appendix was first written. Since then, RFC 3168 ECN has been widely deployed and L4S did not take this approach anyway. So this approach is not discussed further, because it is no longer a feasible option.

B.6. Protocol ID

It has been suggested that a new Protocol ID in the IPv4 Protocol field or the IPv6 Next Header field could identify L4S packets. However this approach is ruled out by numerous problems:

- A duplicate protocol ID would need to be created for each transport (TCP, SCTP, UDP, etc.).

- In IPv6, there can be a sequence of Next Header fields, and it would not be obvious which one would be expected to identify a network service like L4S.

- A new protocol ID would rarely provide an end-to-end service, because It is well-known that new protocol IDs are often blocked by numerous types of middlebox.

- The approach is not a solution for AQM methods below the IP layer.

B.7. Source or destination addressing

Locally, a network operator could arrange for L4S service to be applied based on source or destination addressing, e.g. packets from its own data centre and/or CDN hosts, packets to its business customers, etc. It could use addressing at any layer, e.g. IP addresses, MAC addresses, VLAN IDs, etc. Although addressing might be a useful tactical approach for a single ISP, it would not be a feasible approach to identify an end-to-end service like L4S. Even for a single ISP, it would require packet classifiers in buffers to be dependent on changing topology and address allocation decisions

elsewhere in the network. Therefore this approach is not a feasible solution.

B.8. Summary: Merits of Alternative Identifiers

Table 1 and Table 2 provide a very high level summary of the pros and cons detailed against the schemes described respectively in Appendix B.1, Appendix B.4, Appendix B.2 and Appendix B.3 for nine issues that set them apart.

Issue	ECT(1) + CE (Chosen)		DSCP + ECN	
	initial	eventual	initial	eventual
end-to-end	. . Y	. . Y	N . .	. ? .
tunnels	. . ?	. . Y	. O .	. O .
lower layers	. O .	. . ?	N . .	. ? .
codepoints	N . .	. . ?	N . .	. . ?
reordering	. O .	. . ?	. . Y	. . Y
identify C AQM	. O .	. . ?	. O .	. . ?
L3-only poss.	. . Y	. . Y	. . Y	. . Y
TCP feedback	. O .	. . Y	. O .	. . Y
TCP ctrl pkts	. O .	. . ?	. . Y	. . Y

Table 1: Merits of Alternative L4S Identifiers (pt 1)

Issue	ECN-DualQ-SCE1		ECN-DualQ-SCE0	
	initial	eventual	initial	eventual
end-to-end	. . Y	. . Y	. . Y	. . Y
tunnels	. ? .	. . ?	N . .	? . .
lower layers	N . .	. ? .	N . .	? . .
codepoints	N . .	. ? .	N . .	. ? .
reordering	N . .	N . .	N . .	N . .
identify C AQM	. . Y	. . Y	. . Y	. . Y
L3-only poss	N . .	N . .	. . Y	. . Y
TCP feedback	N . .	. O .	N . .	. O .
TCP ctrl pkts	. O .	. . ?	. O .	. . ?

Table 2: Merits of Alternative L4S Identifiers (pt 2)

The schemes are scored based on both their capabilities now ('initial') and in the long term ('eventual'). The scores are one of

'N, O, Y', meaning 'Poor', 'Ordinary', 'Good' respectively. The same scores are aligned vertically to aid the eye. A score of "?" in one of the positions means that this approach might optimistically become this good, given sufficient effort. The tables summarize the text and are not meant to be understandable without having read the text.

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

The ECT(1) codepoint of the ECN field has already been assigned once for the ECN nonce [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 following categories:

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 protocol [RFC3540] proposed that a TCP sender could set either of 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 for integrity protection in future. The ECN Nonce RFC [RFC3540] as been reclassified as historic, partly because other ways have been developed to protect feedback integrity of TCP and other transports [RFC8311] that do not consume a codepoint in the IP header. For instance:

- the sender can test the integrity 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 para 2 of Section 20.2 of [RFC3168]. This works for loss and it will work for the accurate ECN feedback [RFC7560] intended for L4S.

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

suppressing congestion feedback or the sender is unresponsive to the feedback, or both, ConEx audit can neutralise any advantage that any of these three parties would otherwise gain.

- The TCP authentication option (TCP-AO [RFC5925]) can be used to detect any tampering with TCP congestion feedback (whether malicious or accidental). TCP's congestion feedback fields are immutable end-to-end, so they are amenable to TCP-AO protection, which covers the main TCP header and TCP options by default. However, TCP-AO 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.

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. VCP [VCP], 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 standardisation 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 ECN field. It uses ECT(1) to signify a less severe level of pre-congestion notification than CE [RFC6660]. However, the 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 any of the L4S service identifiers proposed in Appendix B.

Authors' Addresses

Koen De Schepper
Nokia Bell Labs
Antwerp
Belgium

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

Bob Briscoe (editor)
Independent
UK

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