Transport Services (tsv)
Internet-Draft
Intended status: Informational
Expires: December 12, 2016

B. Briscoe, Ed.
Simula Research Lab
K. De Schepper
Nokia Bell Labs
M. Bagnulo Braun
Universidad Carlos III de Madrid
June 10, 2016

Low Latency, Low Loss, Scalable Throughput (L4S) Internet Service: Problem Statement

draft-briscoe-tsvwg-aqm-tcpm-rmcat-l4s-problem-01

Abstract

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

This document explains the underlying problems that have been preventing the Internet from enjoying such performance improvements. It then outlines the parts necessary for a solution and the steps that will be needed to standardize them. It points out opportunities that will open up, and sets out some likely use-cases, including ultra-low latency interaction with cloud processing over the public Internet.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

1.1. The Application Performance Problem

It is increasingly common for _all_ of a user's applications at any one time to require low delay: interactive Web, Web services, voice, conversational video, interactive video, instant messaging, online gaming, remote desktop and cloud-based applications. In the last decade or so, much has been done to reduce propagation delay by placing caches or servers closer to users. However, queuing remains a major, albeit intermittent, component of latency. When present it typically doubles the path delay from that due to the base speed-of-light. Low loss is also important because, for interactive applications, losses translate into even longer retransmission delays.

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

Therefore, the goal is an Internet service with ultra-Low queueing Latency, ultra-Low Loss and Scalable throughput (L4S) - for _all_ traffic. Having motivated the goal of 'L4S for all', this document enumerates the problems that have to be overcome to reach it.

It must be said that queuing delay only degrades performance infrequently [Hohlfeld14]. It only occurs when a large enough capacity-seeking (e.g. TCP) flow is running alongside the user's traffic in the bottleneck link, which is typically in the access network. Or when the low latency application is itself a large capacity-seeking flow (e.g. interactive video). At these times, the performance improvement must be so remarkable that network operators will be motivated to deploy it.

1.2. The Technology Problem

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

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

It has been convincingly demonstrated [DCttH15] that it is possible to deploy such an L4S service alongside the existing best efforts service so that all of a user's applications can shift to it when their stack is updated. Access networks are typically designed with one link as the bottleneck for each site (which might be a home, small enterprise or mobile device), so deployment at a single node should give nearly all the benefit. Although the main incremental deployment problem has been solved, and the remaining work seems straightforward, there may need to be changes in approach during the process of engineering a complete solution.

There are three main parts to the L4S approach (illustrated in Figure 1):

2) Network: The L4S service needs to be isolated from the queuing latency of the Classic service. However, the two should be able to freely share a common pool of capacity. This is because there is no way to predict how many flows at any one time might use each service and capacity in access networks is too scarce to partition into two. So a 'semi-permeable' membrane is needed that partitions latency but not bandwidth. The Dual Queue Coupled AQM [I-D.briscoe-aqm-dualq-coupled] is an example of such a semi-permeable membrane.

   Per-flow queuing such as in [I-D.ietf-aqm-fq-codel] could be used, but it partitions both latency and bandwidth between every e2e

flow. So it is rather overkill, which brings disadvantages (see Section 2.2), not least that thousands of queues are needed when two are sufficient.

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

3) Host: Scalable congestion controls already exist. They solve the scaling problem with TCP first pointed out in [RFC3649]. The one used most widely (in controlled environments) is Data Centre TCP (DCTCP [I-D.ietf-tcpm-dctcp]), which has been implemented and deployed in Windows Server Editions (since 2012), in Linux and in FreeBSD. Although DCTCP as-is 'works' well over the public Internet, most implementations lack certain safety features that will be necessary once it is used outside controlled environments like data centres (see later). A similar scalable congestion control will also need to be transplanted into protocols other than TCP (SCTP, RTP/RTCP, RMCAT, etc.)

```
                       (1)                      (2)
                .-------^------.  .--------------^-------------------.
  ,-(3)-----.                                    ______
 ; ________  :             L4S    --------.     |      |
 :|Scalable| :                _\         ||___\_| mark |
 :| sender | :   __________  / /         ||   / |______|\   _________
 :|________|\;  |          |/     --------'         ^    \1|         |
  `---------'\__|  IP-ECN  |               Coupling :     \|priority |_\
   ________  /  |Classifier|                        :     /|scheduler| /
  |Classic |/   |__________|\     --------.      ___:__  / |_________|
  | sender |                 \_\   || | |||___\_| mark/|/
  |________|                   /   || | |||   / | drop |
                        Classic   --------'     |______|
```

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

1.3. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. In this document, these words will appear with that interpretation only when

in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

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

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

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

Scalable Congestion Control: A congestion control where flow rate is inversely proportional to the level of congestion signals. Then, as flow rate scales, the number of congestion signals per round trip remains invariant, maintaining the same degree of control. For instance, DCTCP averages 2 congestion signals per round-trip whatever the flow rate.

Classic Congestion Control: A congestion control with a flow rate compatible with standard TCP Reno [RFC5681]. With Classic congestion controls, as capacity increases enabling higher flow rates, the number of round trips between congestion signals (losses or ECN marks) rises in proportion to the flow rate. So control of queuing and/or utilization becomes very slack. For instance, with 1500 B packets and an RTT of 18 ms, as TCP Reno flow rate increases from 2 to 100 Mb/s the number of round trips between congestion signals rises proportionately, from 2 to 100.

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

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

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

1.4. The Standardization Problem

0) Architecture: The first step will be to articulate the structure and interworking requirements of the set of parts that would satisfy the overall application performance requirements.

Then specific interworking aspects of the following three components parts will need to be defined:

1) Protocol:

   A. [I-D.briscoe-tsvwg-ecn-l4s-id] recommends ECT(1) is used as the identifier to classify L4S and Classic packets into their separate treatments, as required by [RFC4774]. The draft also points out that the original experimental assignment of this codepoint as an ECN nonce [RFC3540] needs to be made obsolete (it was never deployed, and it offers no security benefit now that deployment is optional).

   B. An essential aspect of a scalable congestion control is the use of explicit congestion signals rather than losses, because the signals need to be sent immediately and frequently--too often to use drops. 'Classic' ECN [RFC3168] requires an ECN signal to be treated the same as a drop, both when it is generated in the network and when it is responded to by hosts. L4S allows networks and hosts to support two separate meanings for ECN. So the standards track [RFC3168] will need to be updated to allow ECT(1) packets to depart from the 'same as drop' constraint.

2) Network: The Dual Queue Coupled AQM has been specified as generically as possible [I-D.briscoe-aqm-dualq-coupled] as a 'semi-permeable' membrane without specifying the particular AQMs to use in the two queues. An informational appendix of the draft is provided for pseudocode examples of different possible AQM approaches. Initially a zero-config variant of RED called Curvy RED was implemented, tested and documented. A variant of PIE has been implemented and tested and is about to be documented. The aim is for designers to be free to implement diverse ideas. So the brief normative body of the draft only specifies the minimum constraints an AQM needs to comply with to ensure that the L4S and Classic services will coexist.

3) Host:

A. Data Centre TCP is the most widely used example of a scalable congestion control. It is being documented in the TCPM WG as an informational record of the protocol currently in use [I-D.ietf-tcpm-dctcp]. It will be necessary to define a number of safety features for a variant usable on the public Internet. A draft list of these, known as the TCP Prague requirements, has been drawn up (see Appendix A).

B. Transport protocols other than TCP use various congestion controls designed to be friendly with Classic TCP. It will be necessary to implement scalable variants of each of these transport behaviours before they can use the L4S service. The following standards track RFCs currently define these protocols, and they will need to be updated to allow a different congestion response, which they will have to indicate by using the ECT(1) codepoint: ECN in TCP [RFC3168], in SCTP [RFC4960], in RTP [RFC6679], and in DCCP [RFC4340].

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

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

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

Currently, the new specification of the ECN protocol [I-D.briscoe-tsvwg-ecn-l4s-id] has been written for the experimental track. Perhaps a better approach would be to make this a standards track protocol draft that updates the definition of ECT(1) in all the above standards track RFCs and obsoletes its experimental use for the ECN nonce. Then experimental specifications of example network (AQM) and host (congestion control) algorithms can be written.

2. Rationale

2.1. Why These Primary Components?

Explicit congestion signalling (protocol): Explicit congestion signalling is a key part of the L4S approach. In contrast, use of drop as a congestion signal creates a tension because drop is both a useful signal (more would reduce delay) and an impairment (less would reduce delay). Explicit congestion signals can be used many times per round trip, to keep tight control, without any impairment. Under heavy load, even more explicit signals can be applied so the queue can be kept short whatever the load. Whereas state-of-the-art AQMs have to introduce very high packet drop at high load to keep the queue short. Further, TCP's sawtooth reduction can be smaller, and therefore return to the operating point more often, without worrying that this causes more signals (one at the top of each smaller sawtooth). The consequent smaller amplitude sawteeth fit between a very shallow marking threshold and an empty queue, so delay variation can be very low, without risk of under-utilization.

All the above makes it clear that explicit congestion signalling is only advantageous for latency if it does not have to be considered 'the same as' drop (as required with Classic ECN [RFC3168]). Before Classic ECN was standardized, there were various proposals to give an ECN mark a different meaning from drop. However, there was no particular reason to agree on any one of the alternative meanings, so 'the same as drop' was the only compromise that could be reached. RFC 3168 contains a statement that:

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

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

Similarly, coupling the congestion notification between the queues is not necessarily essential, but it is a clever and simple way to allow senders to determine their rate, packet-by-packet, rather than be overridden by a network scheduler. Because otherwise a network scheduler would have to inspect at least transport layer

headers, and it would have to continually assign a rate to each flow without any easy way to understand application intent.

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

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

2.2. Why Not Alternative Approaches?

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

Diffserv: Diffserv addresses the problem of bandwidth apportionment for important traffic as well as queuing latency for delay-sensitive traffic. L4S solely addresses the problem of queuing latency. Diffserv will still be necessary where important traffic requires priority (e.g. for commercial reasons, or for protection of critical infrastructure traffic). Nonetheless, if there are Diffserv classes for important traffic, the L4S approach can provide low latency for _all_ traffic within each Diffserv class (including the case where there is only one Diffserv class).

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

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

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

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

B. fq requires packet inspection into the end-to-end transport layer, which doesn't sit well alongside encryption for privacy - in contrast a dual queue only operates at the IP layer;

C. fq decides packet-by-packet which flow to schedule without knowing application intent. In contrast, in the L4S approach the sender still controls the relative rate of each flow dependent on the needs of each application.

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

3. Opportunities

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

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

o Gaming

o VoIP

o Video conferencing

o Web browsing

o (Adaptive) video streaming

o Instant messaging

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

o Cloud based interactive video

o Cloud based virtual and augmented reality

The above two applications have been successfully demonstrated with L4S, both running together over a 40 Mb/s broadband access link loaded up with the numerous other latency sensitive applications in the previous list as well as numerous downloads. A panoramic video of a football stadium can be swiped and pinched so that on the fly a proxy in the cloud generates a sub-window of the match video under the finger-gesture control of each user. At the same time, a virtual reality headset fed from a 360 degree camera in a racing car has been demonstrated, where the user's head movements control the scene generated in the cloud. In both cases, with 7 ms end-to-end base delay, the additional queuing delay of roughly 1 ms is so low that it seems the video is generated locally. See https://riteproject.eu/dctth/ for videos of these demonstrations.

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

If low network delay is not available, all fine interaction has to be done locally and therefore much more redundant data has to be downloaded. When all interactive processing can be done in the cloud, only the data to be rendered for the end user needs to be sent. Whereas, once applications can rely on minimal queues in the network, they can focus on reducing their own latency by only minimizing the application send queue.

3.1. Use Cases

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

o Where the bottleneck is one of various types of access network: DSL, cable, mobile, satellite

   * Radio links (cellular, WiFi) that are distant from the source are particularly challenging. The radio link capacity can vary rapidly by orders of magnitude, so it is often desirable to hold a buffer to utilise sudden increases of capacity;

   * cellular networks are further complicated by a perceived need to buffer in order to make hand-overs imperceptible;

   * Satellite networks generally have a very large base RTT, so even with minimal queuing, overall delay can never be extremely low;

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

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

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

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

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

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

   * elastic (TCP/SCTP);

   * real-time (RTP, RMCAT);

   * query (DNS/LDAP).

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

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

4. IANA Considerations

This specification contains no IANA considerations.

5. Security Considerations

5.1. Traffic (Non-)Policing

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

During early deployment (and perhaps always), some networks will not offer the L4S service. These networks do not need to police or re-mark L4S traffic - they just forward it unchanged as best efforts traffic, as they would already forward traffic with ECT(1) today. At a bottleneck, such networks will introduce some queuing and dropping. When a scalable congestion control detects a drop it will have to respond as if it is a Classic congestion control (see item 3-1 in Appendix A). This will ensure safe interworking with other traffic at the 'legacy' bottleneck.

Certain network operators might choose to restict access to the L4S class, perhaps only to customers who have paid a premium. In the packet classifer (item 2 in Figure 1), they could identify such customers using some other field than ECN (e.g. source address range), and just ignore the L4S identifier for non-paying customers. This would ensure that the L4S identifier survives end-to-end even though the service does not have to be supported at every hop. Such arrangements would only require simple registered/not-registered packet classification, rather than the managed application-specific traffic policing against customer-specific traffic contracts that Diffserv requires.

5.2. 'Latency Friendliness'

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

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

5.3. ECN Integrity

Receiving hosts can fool a sender into downloading faster by suppressing feedback of ECN marks (or of losses if retransmissions are not necessary or available otherwise). [RFC3540] proposes that a TCP sender could pseudorandomly set either of ECT(0) or ECT(1) in each packet of a flow and remember the sequence it had set, termed the ECN nonce. If the receiver supports the nonce, it can prove that it is not suppressing feedback by reflecting its knowledge of the sequence back to the sender. The nonce was proposed on the assumption that receivers might be more likely to cheat congestion control than senders (although senders also have a motive to cheat).

If L4S uses the ECT(1) codepoint of ECN for packet classification, it will have to obsolete the experimental nonce. As far as is known, the ECN Nonce has never been deployed, and it was only implemented for a couple of testbed evaluations. It would be nearly impossible to deploy now, because any misbehaving receiver can simply opt-out, which would be unremarkable given all receivers currently opt-out.

Other ways to protect TCP feedback integrity have since been developed. 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 [I-D.moncaster-tcpm-rcv-cheat]. This method consumes no extra codepoints. It works for loss and it will work for ECN feedback in any transport protocol suitable for L4S. However, it shares the same assumption as the nonce; that the sender is not cheating and it is motivated to prevent the receiver cheating;

- 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. ConEx is only currently defined for IPv6 and consumes a destination option header. It has been implemented, but not deployed as far as is known.

6. Acknowledgements

7. References

7.1. 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, <http://www.rfc-editor.org/info/rfc2119>.

7.2. Informative References

[DCttH15] De Schepper, K., Bondarenko, O., Briscoe, B., and I. Tsang, "'Data Centre to the Home': Ultra-Low Latency for All", 2015, <http://www.bobbriscoe.net/projects/latency/dctth_preprint.pdf>.

(Under submission)

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

[I-D.briscoe-aqm-dualq-coupled] Schepper, K., Briscoe, B., Bondarenko, O., and I. Tsang, "DualQ Coupled AQM for Low Latency, Low Loss and Scalable Throughput", draft-briscoe-aqm-dualq-coupled-01 (work in progress), March 2016.

[I-D.briscoe-tsvwg-ecn-l4s-id] Schepper, K., Briscoe, B., and I. Tsang, "Identifying Modified Explicit Congestion Notification (ECN) Semantics for Ultra-Low Queuing Delay", draft-briscoe-tsvwg-ecn-l4s-id-01 (work in progress), March 2016.

[I-D.ietf-aqm-fq-codel] Hoeiland-Joergensen, T., McKenney, P., dave.taht@gmail.com, d., Gettys, J., and E. Dumazet, "The FlowQueue-CoDel Packet Scheduler and Active Queue Management Algorithm", draft-ietf-aqm-fq-codel-06 (work in progress), March 2016.

[I-D.ietf-aqm-pie]
Pan, R., Natarajan, P., Baker, F., and G. White, "PIE: A Lightweight Control Scheme To Address the Bufferbloat Problem", draft-ietf-aqm-pie-08 (work in progress), June 2016.

[I-D.ietf-tcpm-accurate-ecn]
Briscoe, B., KĂźhlewind, M., and R. Scheffenegger, "More Accurate ECN Feedback in TCP", draft-ietf-tcpm-accurate-ecn-00 (work in progress), December 2015.

[I-D.ietf-tcpm-cubic]
Rhee, I., Xu, L., Ha, S., Zimmermann, A., Eggert, L., and R. Scheffenegger, "CUBIC for Fast Long-Distance Networks", draft-ietf-tcpm-cubic-01 (work in progress), January 2016.

[I-D.ietf-tcpm-dctcp]
Bensley, S., Eggert, L., Thaler, D., Balasubramanian, P., and G. Judd, "Datacenter TCP (DCTCP): TCP Congestion Control for Datacenters", draft-ietf-tcpm-dctcp-01 (work in progress), November 2015.

[I-D.khademi-tcpm-alternativebackoff-ecn]
Khademi, N., Welzl, M., Armitage, G., and G. Fairhurst, "TCP Alternative Backoff with ECN (ABE)", draft-khademi-tcpm-alternativebackoff-ecn-00 (work in progress), May 2016.

[I-D.moncaster-tcpm-rcv-cheat]
Moncaster, T., Briscoe, B., and A. Jacquet, "A TCP Test to Allow Senders to Identify Receiver Non-Compliance", draft-moncaster-tcpm-rcv-cheat-03 (work in progress), July 2014.

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

[I-D.you-encrypted-traffic-management]
You, J. and C. Xiong, "The Effect of Encrypted Traffic on the QoS Mechanisms in Cellular Networks", draft-you-encrypted-traffic-management-00 (work in progress), October 2015.

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

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

[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, <http://www.rfc-editor.org/info/rfc3168>.

[RFC3246] Davie, B., Charny, A., Bennet, J., Benson, K., Le Boudec, J., Courtney, W., Davari, S., Firoiu, V., and D. Stiliadis, "An Expedited Forwarding PHB (Per-Hop Behavior)", RFC 3246, DOI 10.17487/RFC3246, March 2002, <http://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, <http://www.rfc-editor.org/info/rfc3540>.

[RFC3649] Floyd, S., "HighSpeed TCP for Large Congestion Windows", RFC 3649, DOI 10.17487/RFC3649, December 2003, <http://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, <http://www.rfc-editor.org/info/rfc4340>.

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

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

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

[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, <http://www.rfc-editor.org/info/rfc6679>.

[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, <http://www.rfc-editor.org/info/rfc7560>.

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

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

[TCPPrague] Briscoe, B., "Notes: DCTCP evolution 'bar BoF': Tue 21 Jul 2015, 17:40, Prague", tcpprague mailing list archive , July 2015.

Appendix A. The "TCP Prague Requirements"

This list of requirements was produced at an ad hoc meeting during IETF-94 in Prague [TCPPrague]. The list prioritised features that would need to be added to DCTCP to make it safe for use on the public Internet alongside existing non-DCTCP traffic (up to #3-7). After that, it also includes features to improve the performance of DCTCP in the wider range of conditions found on the public Internet.

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

The references should be consulted for why each requirement is considered necessary for safety. There follow brief reasons for those that are not self-explanatory and have not yet been written up:

#3-5 Reduce RTT-dependence: Classic TCP's throughput is known to be inversely proportional to RTT. One would expect flows over very low RTT paths to nearly starve flows over larger RTTs. However,

because Classic TCP induces a large queue, it has never allowed a very low RTT path to exist, so far. For instance, consider two paths with base RTT 1ms and 100ms. If Classic TCP induces a 20ms queue, it turns these RTTs into 21ms and 120ms leading to a throughput ratio of about 1:6. Whereas if a Scalable TCP induces only a 1ms queue, the ratio is 2:101. Therefore, with small queues, long RTT flows will essentially starve.

Smooth ECN feedback over own RTT: DCTCP currently smooths feedback over a hard-coded number of segments, with the value optimized for data centres. For the wider range of round-trip times on the public Internet, it needs to smooth over roughly one window of packets. Otherwise it could respond too rapidly (or too sluggishly) and become unstable (or unresponsive).

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

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

TCP: Applicable to all forms of TCP congestion control;

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

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

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

Req #	Requirement	Reference
0	ARCHITECTURE	
1	L4S IDENTIFIER	[I-D.briscoe-tsvwg-ecn-l4s-id]
2	DUAL QUEUE AQM	[I-D.briscoe-aqm-dualq-coupled]
	SCALABLE TRANSPORT - SAFETY ADDITIONS	
3-1	Fall back to Reno/Cubic on loss	[I-D.ietf-tcpm-dctcp]
3-2	Suitable ECN Feedback	[I-D.ietf-tcpm-accurate-ecn], [I-D.stewart-tsvwg-sctpecn].
3-4	Scaling TCP's Congestion Window for Small Round Trip Times	[TCP-sub-mss-w]
3-5	Reduce RTT-dependence	
3-6	Smooth ECN feedback over own RTT	
3-7	Fall back to Reno/Cubic if classic ECN bottleneck detected	
	SCALABLE TRANSPORT - PERFORMANCE ENHANCEMENTS	
3-8	Faster-than-additive increase	
3-9	Less drastic exit from slow-start	

#	WG	TCP	DCTCP	DCTCP-bis	TCP Prague	SCTP Prague	RMCAT Prague
0	tsvwg?	Y	Y	Y	Y	Y	Y
1	tsvwg?			Y	Y	Y	Y
2	aqm?	n/a	n/a	n/a	n/a	n/a	n/a
3-1	tcpm		Y	Y	Y	Y	Y
3-2	tcpm	Y	Y	Y	Y	n/a	n/a
3-4	tcpm	Y	Y	Y	Y	Y	?
3-5	tcpm/ iccrg?			Y	Y	Y	?
3-6	tcpm/ iccrg?		?	Y	Y	Y	?
3-7	tcpm/ iccrg?				Y	Y	?
3-8	tcpm/ iccrg?			Y	Y	Y	?
3-9	tcpm/ iccrg?			Y	Y	Y	?

Authors' Addresses

Bob Briscoe (editor)
Simula Research Lab

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

Koen De Schepper
Nokia Bell Labs
Antwerp
Belgium

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

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

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