Transport Area Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 30, 2019

G. White
CableLabs
T. Fossati
ARM

June 28, 2019

Identifying and Handling Non Queue Building Flows in a Bottleneck Link
draft-white-tsvwg-nqb-02

Abstract

This draft proposes the definition of a standardized DiffServ code point (DSCP) to identify Non-Queue-Building flows (for example: interactive voice and video, gaming, machine to machine applications), along with a Per-Hop-Behavior (PHB) that provides a separate queue for such flows.

The purpose of such a marking scheme is to enable networks to provide and utilize queues that are optimized to provide low latency and low loss for such Non-Queue-Building flows (e.g. shallow buffers, optimized media access parameters, etc.).

This marking scheme and PHB has been developed primarily for use by access network segments, where queuing delays and queuing loss caused by Queue-Building protocols are manifested. In particular, applications to cable broadband links and mobile network radio and core segments are discussed.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 30, 2019.

Table of Contents

1. Introduction

The vast majority of packets that are carried by broadband access networks are managed by an end-to-end congestion control algorithm, such as Reno, Cubic or BBR. These congestion control algorithms attempt to seek the available capacity of the end-to-end path (which can frequently be the access network link capacity), and in doing so generally overshoot the available capacity, causing a queue to build-up at the bottleneck link. This queue build up results in queuing delay that the application experiences as variable latency, and commonly results in packet loss as well.

In contrast to traditional congestion-controlled applications, there are a variety of relatively low data rate applications that do not materially contribute to queueing delay and loss, but are nonetheless subjected to it by sharing the same bottleneck link in the access network. Many of these applications may be sensitive to latency or latency variation, as well as packet loss, and thus produce a poor quality of experience in such conditions.

Active Queue Management (AQM) mechanisms (such as PIE [RFC8033], DOCSIS-PIE [RFC8034], or CoDel [RFC8289]) can improve the quality of experience for latency sensitive applications, but there are practical limits to the amount of improvement that can be achieved without impacting the throughput of capacity-seeking applications.

This document considers differentiating between these two classes of traffic in bottleneck links and queuing them separately in order that both classes can deliver optimal quality of experience for their applications.

A couple of preconditions need to be satisfied before we can move on from the status quo. First, the packets must be efficiently identified so that they can be quickly assigned to the "right" queue. This is especially important with the rising popularity of encrypted and multiplexed transports, which has the potential of making deep inspection infeasible. Second, the signal must be such that malicious or badly configured nodes can't abuse it.

2. Requirements Language

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 RFC 2119 [RFC2119].

3. Non-Queue Building Flows

There are many applications that send traffic at relatively low data rates and/or in a fairly smooth and consistent manner such that they are highly unlikely to exceed the available capacity of the network path between source and sink. These applications do not make use of network buffers, but nonetheless can be subjected to packet delay and delay variation as a result of sharing a network buffer with those that do make use of them. Many of these applications are negatively affected by excessive packet delay and delay variation. Such applications are ideal candidates to be queued separately from the capacity-seeking applications that are the cause of queue buildup, latency and loss.

These Non-queue-building (NQB) flows are typically UDP flows that send traffic at a lower data rate and don't seek the capacity of the link (examples: online games, voice chat, DNS lookups). Here the data rate is essentially limited by the Application itself. In contrast, Queue-building (QB) flows include traffic which uses the Traditional TCP or QUIC, with BBR or other TCP congestion controllers.

There are a lot of great examples of applications that fall very neatly into these two categories, but there are also application flows that may be in a gray area in between (e.g. they are NQB on higher-speed links, but QB on lower-speed links).

4. Endpoint Marking and Queue Protection

This memo proposes that application endpoints apply a marking, utilizing the Diffserv field of the IP header, to packets of NQB flows that could then be used by the network to differentiate between QB and NQB flows. It is important for such a marking to be universally agreed upon, rather than being locally defined by the network operator, such that applications could be written to apply the marking without regard to local network policies.

Some questions that arise when considering endpoint marking are: How can an application determine whether it is queue building or not, given that the sending application is generally not aware of the available capacity of the path to the receiving endpoint? Even in cases where an application is aware of the capacity of the path, how can it be sure that the available capacity (considering other flows that may be sharing the path) would be sufficient to result in the application's traffic not causing a queue to form? In an unmanaged environment, how can networks trust endpoint marking, and why wouldn't all applications mark their packets as NQB?

As an answer the last question, it is worthwhile to note that the NQB designation and marking would be intended to convey verifiable traffic behavior, not needs or wants. Also, it would be important that incentives are aligned correctly, i.e. that there is a benefit to the application in marking its packets correctly, and no benefit for an application in intentionally mismarking its traffic. Thus, a useful property of nodes that support separate queues for NQB and QB flows would be that for NQB flows, the NQB queue provides better performance (considering latency, loss and throughput) than the QB queue; and for QB flows, the QB queue provides better performance (considering latency, loss and throughput) than the NQB queue.

Even so, it is possible that due to an implementation error or misconfiguration, a QB flow would end up getting mismarked as NQB, or

vice versa. In the case of an NQB flow that isn't marked as NQB and ends up in the QB queue, it would only impact its own quality of service, and so it seems to be of lesser concern. However, a QB flow that is mismarked as NQB would cause queuing delays for all of the other flows that are sharing the NQB queue.

To prevent this situation from harming the performance of the real NQB flows, network elements that support differentiating NQB traffic SHOULD support a "queue protection" function that can identify QB flows that are mismarked as NQB, and reclassify those flows/packets to the QB queue. This benefits the reclassified flow by giving it access to a large buffer (and thus lower packet loss rate), and benefits the actual NQB flows by preventing harm (increased latency variability) to them. Such a function SHOULD be implemented in an objective and verifiable manner, basing its decisions upon the behavior of the flow rather than on application-layer constructs.

5. Non Queue Building PHB and DSCP

This section uses the DiffServ nomenclature of per-hop-behavior (PHB) to describe how a network node could provide better quality of service for NQB flows without reducing performance of QB flows.

A node supporting the NQB PHB MUST provide a queue for non-queue-building traffic separate from the queue used for queue-building traffic. This queue SHOULD support a latency-based queue protection mechanism that is able to identify queue-building behavior in flows that are classified into the queue, and to redirect flows causing queue build-up to a different queue. One example algorithm can be found in Annex P of [DOCSIS-MULPIv3.1].

While there may be some similarities between the characteristics of NQB flows and flows marked with the Expedited Forwarding (EF) DSCP, the NQB PHB would differ from the Expedited Forwarding PHB in several important ways.

- NQB traffic is not rate limited or rate policed. Rather, the NQB queue would be expected to support a latency-based queue protection mechanism that identifies NQB marked flows that are beginning to cause latency, and redirects packets from those flows to the queue for QB flows.

- The node supporting the NQB PHB makes no guarantees on latency or data rate for NQB marked flows, but instead aims to provide a bound on queuing delay for as many such marked flows as it can, and shed load when needed.

- o EF is commonly used exclusively for voice traffic, for which additional functions are applied, such as admission control, accounting, prioritized delivery, etc.

In networks that support the NQB PHB, it may be preferred to also include traffic marked EF (0b101110) in the NQB queue. The choice of the 0x2A codepoint (0b101010) for NQB would conveniently allow a node to select these two codepoints using a single mask pattern of 0b101x10.

6. End-to-end Support

In contrast to the existing standard DSCPs, which are typically only meaningful within a DiffServ Domain (e.g. an AS), this DSCP would be intended for end-to-end usage across the Internet. Some network operators bleach the Diffserv field on ingress into their network [Custura], and in some cases apply their own DSCP for internal usage. Networks that support the NQB PHB SHOULD preserve the NQB DSCP when forwarding via an interconnect.

7. Relationship to L4S

The dual-queue mechanism described in this draft is intended to be compatible with [I-D.ietf-tsvwg-l4s-arch].

8. Use Cases

8.1. DOCSIS Access Networks

Residential cable broadband Internet services are commonly configured with a single bottleneck link (the access network link) upon which the service definition is applied. The service definition, typically an upstream/downstream data rate tuple, is implemented as a configured pair of rate shapers that are applied to the user's traffic. In such networks, the quality of service that each application receives, and as a result, the quality of experience that it generates for the user is influenced by the characteristics of the access network link.

To support the NQB PHB, cable broadband services MUST be configured to provide a separate queue for NQB traffic that shares the service rate shaping configuration with the queue for QB traffic.

8.2. Mobile Networks

Today's mobile networks are configured to bundle all flows to and from the Internet into a single "default" EPS bearer whose buffering characteristics are not compatible with low-latency traffic. The

established behaviour is partly rooted in the desire to prioritise operators' voice services over competing over-the-top services. Of late, said business consideration seems to have lost momentum and the incentives might now be aligned towards allowing a more suitable treatment of Internet real-time flows.

To support the NQB PHB, the mobile network MUST be configured to give UEs a dedicated, low-latency, non-GBR, EPS bearer with QCI 7 in addition to the default EPS bearer.

A packet carrying the NQB DSCP SHOULD be routed through the dedicated low-latency EPS bearer. A packet that has no associated NQB marking SHOULD be routed through the default EPS bearer.

8.3. WiFi Networks

WiFi networking equipment compliant with 802.11e generally supports either four or eight transmit queues and four sets of associated CSMA parameters that are used to enable differentiated media access characteristics. Implementations typically utilize the IP DSCP field to select a transmit queue.

As discussed in [RFC8325], most implementations use a default DSCP to User Priority mapping that utilizes the most significant three bits of the DiffServ Field to select User Priority. In the case of the 0x2A codepoint, this would map to UP_5 which is in the "Video" Access Category (one level above "Best Effort").

Systems that utilize [RFC8325], SHOULD map the 0x2A codepoint to UP_6 in the "Voice" Access Category.

9. Comparison to Existing Approaches

Traditional QoS mechanisms focus on prioritization in an attempt to achieve two goals: reduced latency for "latency-sensitive" traffic, and increased bandwidth availability for "important" applications. Applications are generally given priority in proportion to some combination of latency-sensitivity and importance.

Downsides to this approach include the difficulties in sorting out what priority level each application should get (making the value judgement as to relative latency-sensitivity and importance), associating packets to priority levels (configuring and maintaining lots of classifier state, or trusting endpoint markings and the value judgements that they convey), ensuring that high priority traffic doesn't starve lower priority traffic (admission control, weighted scheduling, etc. are possible solutions). This solution can work in a managed network, where the network operator can control the usage

of the QoS mechanisms, but has not been adopted end-to-end across the Internet. See also [Claffy] for an exhaustive treatment of the argument.

Flow queueing (FQ) approaches (such as fq_codel [RFC8290]), on the other hand, achieve latency improvements by associating packets into "flow" queues and then prioritizing "sparse flows", i.e. packets that arrive to an empty flow queue. Flow queueing does not attempt to differentiate between flows on the basis of value (importance or latency-sensitivity), it simply gives preference to sparse flows, and tries to guarantee that the non-sparse flows all get an equal share of the remaining channel capacity and are interleaved with one another. As a result, FQ mechanisms could be considered more appropriate for unmanaged environments and general Internet traffic.

Downsides to this approach include loss of low latency performance due to the possibility of hash collisions (where a sparse flow shares a queue with a bulk data flow), complexity in managing a large number of queues in certain implementations, and some undesirable effects of the Deficit Round Robin (DRR) scheduling. The DRR scheduler enforces that each non-sparse flow gets an equal fraction of link bandwidth, which causes problems with VPNs and other tunnels, exhibits poor behavior with less-aggressive congestion control algorithms, e.g. LEDBAT [RFC6817], and could exhibit poor behavior with RTP Media Congestion Avoidance Techniques (RMCAT) [I-D.ietf-rmcat-cc-requirements]. In effect, the network element is making a decision as to what constitutes a flow, and then forcing all such flows to take equal bandwidth at every instant.

The Dual-queue approach defined in this document achieves the main benefit of fq_codel: latency improvement without value judgements, without the downsides.

The distinction between NQB flows and QB flows is similar to the distinction made between "sparse flow queues" and "non-sparse flow queues" in fq_codel. In fq_codel, a flow queue is considered sparse if it is drained completely by each packet transmission, and remains empty for at least one cycle of the round robin over the active flows (this is approximately equivalent to saying that it utilizes less than its fair share of capacity). While this definition is convenient to implement in fq_codel, it isn't the only useful definition of sparse flows.

The Linux Heavy-Hitter Filter [HHF][Estan] qdisc and the Cisco Dynamic Packet Prioritization [DPP] feature both categorize application flows into "mice" and "elephants", and provide a separate queue that gives high priority to the "mice" flows. In both of these implementations, the definition of a mice flow is one that falls

below a defined number of bytes or packets (respectively). In essence, the first N bytes or packets of every new flow are queued separately, and given priority over other traffic. The HHF implementation defaults to using 128KB for N, whereas the DPP documentation discusses using 120 packets.

This approach is relatively simple to implement, but it is making the wrong distinction between flows. To illustrate, an hour-long 60 kbps multiplayer online gaming flow sending 60 packets per second would be classified as an elephant after the first 17 seconds using HFF or 2 seconds using DPP, whereas it should be considered as NQB for the entire duration.

Other dual-queue approaches have been proposed, including some that pair a shallow buffer with a deep buffer, similar to what is described in this draft. One such design is the "RD" mechanism in [Podlesny] which proposes that applications select either high rate or low delay, with one queue (the high-rate queue) being given a large buffer and a higher scheduling weight, and the other queue (the low-delay queue) being given a short buffer and lower scheduling weight. This approach is somewhat similar to the NQB PHB, in regards to allowing the application to select between a deep buffer and a shallow one, but it places unnecessary restrictions on the scheduling between the two queues, and doesn't differentiate traffic based on behavior. Further, the approach doesn't provide any safety valve to prevent malicious or misconfigured flows from causing excessive packet loss in the low delay queue. Similarly, the "Loss-Latency Tradeoff" approach described in [I-D.fossati-tsvwg-lola] posits that applications should choose between a queue that provides low latency and potentially high loss (i.e. a shallow buffer), and one that provides low loss and potentially high latency (i.e. a deep buffer). This approach misses that both queuing latency and queuing loss are primarily byproducts of application sending behavior, and by properly segregating applications, no trade-off needs to be made.

10. Acknowledgements

Thanks to Bob Briscoe, Greg Skinner, Dave Taht, Toke Hoeiland-Joergensen and Luca Muscariello for their review comments.

11. IANA Considerations

This draft proposes the registration of a standardized DSCP = 0x2A to denote Non-Queue-Building behavior.

12. Security Considerations

There is no incentive for an application to mismark its packets as NQB (or vice versa). If a queue-building flow were to mark its packets as NQB, it could experience excessive packet loss (in the case that queue-protection is not supported by a node) or it could receive no benefit (in the case that queue-protection is supported). If a non-queue-building flow were to fail to mark its packets as NQB, it could suffer the latency and loss typical of sharing a queue with capacity seeking traffic.

The NQB signal is not integrity protected and could be flipped by an on-path attacker. This might negatively affect the QoS of the tampered flow.

13. Informative References

[Claffy] Claffy, KC. and D. Clark, "Adding Enhanced Services to the Internet: Lessons from History", TPRC , 2015, <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2587262>.

[Custura] Custura, A., Venne, A., and G. Fairhurst, "Exploring DSCP modification pathologies in mobile edge networks", TMA , 2017.

[DOCSIS-MULPIv3.1] Cable Television Laboratories, Inc., "MAC and Upper Layer Protocols Interface Specification, CM-SP-MULPIv3.1-I18-190422", April 22, 2019, <https://specification-search.cablelabs.com/CM-SP-MULPIv3.1>.

[DPP] Cisco, "Intelligent Buffer Management on Cisco Nexus 9000 Series Switches White Paper", June 2017, <https://www.cisco.com/c/en/us/products/collateral/switches/nexus-9000-series-switches/white-paper-c11-738488.html>.

[Estan] Estan, C. and G. Varghese, "New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice", ACM Transactions on Computer Systems Vol.23, Iss.3, August 2003, <https://dl.acm.org/citation.cfm?id=859719>.

[HHF] Lam, T., "net-qdisc-hhf: Heavy-Hitter Filter (HHF) qdisc", December 2013, <https://lwn.net/Articles/577208/>.

[I-D.fossati-tsvwg-lola]
Fossati, T., Fairhurst, G., Gutierrez, P., and M. Kuehlewind, "A Loss-Latency Trade-off Signal for the Mobile Network", draft-fossati-tsvwg-lola-00 (work in progress), December 2018.

[I-D.ietf-rmcat-cc-requirements]
Jesup, R. and Z. Sarker, "Congestion Control Requirements for Interactive Real-Time Media", draft-ietf-rmcat-cc-requirements-09 (work in progress), December 2014.

[I-D.ietf-tsvwg-l4s-arch]
Briscoe, B., Schepper, K., and M. Bagnulo, "Low Latency, Low Loss, Scalable Throughput (L4S) Internet Service: Architecture", draft-ietf-tsvwg-l4s-arch-03 (work in progress), October 2018.

[Podlesny]
Podlesny, M. and S. Gorinsky, "Rd Network Services: Differentiation Through Performance Incentives", SIGCOMM , 2008, <http://people.networks.imdea.org/~sergey_gorinsky/pdf/RD_Services_SIGCOMM_2008.pdf>.

[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>.

[RFC6817] Shalunov, S., Hazel, G., Iyengar, J., and M. Kuehlewind, "Low Extra Delay Background Transport (LEDBAT)", RFC 6817, DOI 10.17487/RFC6817, December 2012, <https://www.rfc-editor.org/info/rfc6817>.

[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>.

[RFC8034] White, G. and R. Pan, "Active Queue Management (AQM) Based on Proportional Integral Controller Enhanced PIE) for Data-Over-Cable Service Interface Specifications (DOCSIS) Cable Modems", RFC 8034, DOI 10.17487/RFC8034, February 2017, <https://www.rfc-editor.org/info/rfc8034>.

[RFC8289] Nichols, K., Jacobson, V., McGregor, A., Ed., and J. Iyengar, Ed., "Controlled Delay Active Queue Management", RFC 8289, DOI 10.17487/RFC8289, January 2018, <https://www.rfc-editor.org/info/rfc8289>.

[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>.

[RFC8325] Szigeti, T., Henry, J., and F. Baker, "Mapping Diffserv to IEEE 802.11", RFC 8325, DOI 10.17487/RFC8325, February 2018, <https://www.rfc-editor.org/info/rfc8325>.

Authors' Addresses

Greg White
CableLabs

Email: g.white@cablelabs.com

Thomas Fossati
ARM

Email: Thomas.Fossati@arm.com