

Network Working Group
Internet-Draft
Obsoletes: 2309 (if approved)
Intended status: Best Current Practice
Expires: August 29, 2015

F. Baker, Ed.
Cisco Systems
G. Fairhurst, Ed.
University of Aberdeen
February 25, 2015

IETF Recommendations Regarding Active Queue Management
draft-ietf-aqm-recommendation-11

Abstract

This memo presents recommendations to the Internet community concerning measures to improve and preserve Internet performance. It presents a strong recommendation for testing, standardization, and widespread deployment of active queue management (AQM) in network devices, to improve the performance of today's Internet. It also urges a concerted effort of research, measurement, and ultimate deployment of AQM mechanisms to protect the Internet from flows that are not sufficiently responsive to congestion notification.

The note replaces the recommendations of RFC 2309 based on fifteen years of experience and new research.

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 <http://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 August 29, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	3
1.1. Congestion Collapse	3
1.2. Active Queue Management to Manage Latency	4
1.3. Document Overview	5
1.4. Changes to the recommendations of RFC2309	6
1.5. Requirements Language	6
2. The Need For Active Queue Management	6
2.1. AQM and Multiple Queues	10
2.2. AQM and Explicit Congestion Marking (ECN)	10
2.3. AQM and Buffer Size	11
3. Managing Aggressive Flows	11
4. Conclusions and Recommendations	14
4.1. Operational deployments SHOULD use AQM procedures	15
4.2. Signaling to the transport endpoints	16
4.2.1. AQM and ECN	17
4.3. AQM algorithm deployment SHOULD NOT require operational tuning	18
4.4. AQM algorithms SHOULD respond to measured congestion, not application profiles.	19
4.5. AQM algorithms SHOULD NOT be dependent on specific transport protocol behaviours	20
4.6. Interactions with congestion control algorithms	21
4.7. The need for further research	22
5. IANA Considerations	23
6. Security Considerations	23
7. Privacy Considerations	23

8. Acknowledgements	24
9. References	24
9.1. Normative References	24
9.2. Informative References	25
Appendix A. Change Log	28
Authors' Addresses	30

1. Introduction

The Internet protocol architecture is based on a connectionless end-to-end packet service using the Internet Protocol, whether IPv4 [RFC0791] or IPv6 [RFC2460]. The advantages of its connectionless design: flexibility and robustness, have been amply demonstrated. However, these advantages are not without cost: careful design is required to provide good service under heavy load. In fact, lack of attention to the dynamics of packet forwarding can result in severe service degradation or "Internet meltdown". This phenomenon was first observed during the early growth phase of the Internet in the mid 1980s [RFC0896][RFC0970], and is technically called "congestion collapse" and was a key focus of RFC2309.

Although wide-scale congestion collapse is not common in the Internet, the presence of localised congestion collapse is by no means rare. It is therefore important to continue to avoid congestion collapse.

Since 1998, when RFC2309 was written, the Internet has become used for a variety of traffic. In the current Internet, low latency is extremely important for many interactive and transaction-based applications. The same type of technology that RFC2309 advocated for combating congestion collapse is also effective at limiting delays to reduce the interaction delay (latency) experienced by applications [Bri15]. High or unpredictable latency can impact the performance of the control loops used by end-to-end protocols (including congestion control algorithms using TCP). There is now also a focus on reducing network latency using the same technology.

The mechanisms described in this document may be implemented in network devices on the path between end-points that include routers, switches, and other network middleboxes. The methods may also be implemented in the networking stacks within endpoint devices that connect to the network.

1.1. Congestion Collapse

The original fix for Internet meltdown was provided by Van Jacobsen. Beginning in 1986, Jacobsen developed the congestion avoidance mechanisms [Jacobson88] that are now required for implementations of

the Transport Control Protocol (TCP) [RFC0793] [RFC1122]. ([RFC7414] provides a roadmap to help identify TCP-related documents.) These mechanisms operate in Internet hosts to cause TCP connections to "back off" during congestion. We say that TCP flows are "responsive" to congestion signals (i.e., packets that are dropped or marked with explicit congestion notification [RFC3168]). It is primarily these TCP congestion avoidance algorithms that prevent the congestion collapse of today's Internet. Similar algorithms are specified for other non-TCP transports.

However, that is not the end of the story. Considerable research has been done on Internet dynamics since 1988, and the Internet has grown. It has become clear that the congestion avoidance mechanisms [RFC5681], while necessary and powerful, are not sufficient to provide good service in all circumstances. Basically, there is a limit to how much control can be accomplished from the edges of the network. Some mechanisms are needed in network devices to complement the endpoint congestion avoidance mechanisms. These mechanisms may be implemented in network devices.

1.2. Active Queue Management to Manage Latency

Internet latency has become a focus of attention to increase the responsiveness of Internet applications and protocols. One major source of delay is the build-up of queues in network devices. Queueing occurs whenever the arrival rate of data at the ingress to a device exceeds the current egress rate. Such queueing is normal in a packet-switched network and is often necessary to absorb bursts in transmission and perform statistical multiplexing of traffic, but excessive queueing can lead to unwanted delay, reducing the performance of some Internet applications.

RFC 2309 introduced the concept of "Active Queue Management" (AQM), a class of technologies that, by signaling to common congestion-controlled transports such as TCP, manages the size of queues that build in network buffers. RFC 2309 also describes a specific AQM algorithm, Random Early Detection (RED), and recommends that this be widely implemented and used by default in routers.

With an appropriate set of parameters, RED is an effective algorithm. However, dynamically predicting this set of parameters was found to be difficult. As a result, RED has not been enabled by default, and its present use in the Internet is limited. Other AQM algorithms have been developed since RFC2309 was published, some of which are self-tuning within a range of applicability. Hence, while this memo continues to recommend the deployment of AQM, it no longer recommends that RED or any other specific algorithm is used as a default; instead it provides recommendations on how to select appropriate

algorithms and that a recommended algorithm is able to automate any required tuning for common deployment scenarios.

Deploying AQM in the network can significantly reduce the latency across an Internet path and since writing RFC2309, this has become a key motivation for using AQM in the Internet. In the context of AQM, it is useful to distinguish between two related classes of algorithms: "queue management" versus "scheduling" algorithms. To a rough approximation, queue management algorithms manage the length of packet queues by marking or dropping packets when necessary or appropriate, while scheduling algorithms determine which packet to send next and are used primarily to manage the allocation of bandwidth among flows. While these two mechanisms are closely related, they address different performance issues and operate on different timescales. Both may be used in combination.

1.3. Document Overview

The discussion in this memo applies to "best-effort" traffic, which is to say, traffic generated by applications that accept the occasional loss, duplication, or reordering of traffic in flight. It also applies to other traffic, such as real-time traffic that can adapt its sending rate to reduce loss and/or delay. It is most effective when the adaption occurs on time scales of a single Round Trip Time (RTT) or a small number of RTTs, for elastic traffic [RFC1633].

Two performance issues are highlighted:

The first issue is the need for an advanced form of queue management that we call "Active Queue Management", AQM. Section 2 summarizes the benefits that active queue management can bring. A number of AQM procedures are described in the literature, with different characteristics. This document does not recommend any of them in particular, but does make recommendations that ideally would affect the choice of procedure used in a given implementation.

The second issue, discussed in Section 4 of this memo, is the potential for future congestion collapse of the Internet due to flows that are unresponsive, or not sufficiently responsive, to congestion indications. Unfortunately, while scheduling can mitigate some of the side-effects of sharing a network queue with an unresponsive flow, there is currently no consensus solution to controlling the congestion caused by such aggressive flows. Methods such as congestion exposure (ConEx) [RFC6789] offer a framework [CONEX] that can update network devices to alleviate these effects. Significant research and engineering will be required before any solution will be available. It is imperative that work to mitigate the impact of

unresponsive flows is energetically pursued, to ensure acceptable performance and the future stability of the Internet.

Section 4 concludes the memo with a set of recommendations to the Internet community on the use of AQM and recommendations for defining AQM algorithms.

1.4. Changes to the recommendations of RFC2309

This memo replaces the recommendations in [RFC2309], which resulted from past discussions of end-to-end performance, Internet congestion, and RED in the End-to-End Research Group of the Internet Research Task Force (IRTF). It follows experience with this and other algorithms, and the AQM discussion within the IETF [AQM-WG].

While RFC2309 described AQM in terms of the length of a queue. This memo changes this, to use AQM to refer to any method that allows network devices to control either the queue length and/or the mean time that a packet spends in a queue.

This memo also explicitly obsoletes the recommendation that Random Early Detection (RED) was to be used as the default AQM mechanism for the Internet. This is replaced by a detailed set of recommendations for selecting an appropriate AQM algorithm. As in RFC2309, this memo also motivates the need for continued research, but clarifies the research with examples appropriate at the time that this memo is published.

1.5. 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 [RFC2119].

2. The Need For Active Queue Management

Active Queue Management (AQM) is a method that allows network devices to control the queue length or the mean time that a packet spends in a queue. Although AQM can be applied across a range of deployment environments, the recommendations in this document are directed to use in the general Internet. It is expected that the principles and guidance are also applicable to a wide range of environments, but may require tuning for specific types of link/network (e.g. to accommodate the traffic patterns found in data centres, the challenges of wireless infrastructure, or the higher delay encountered on satellite Internet links). The remainder of this section identifies the need for AQM and the advantages of deploying AQM methods.

The traditional technique for managing the queue length in a network device is to set a maximum length (in terms of packets) for each queue, accept packets for the queue until the maximum length is reached, then reject (drop) subsequent incoming packets until the queue decreases because a packet from the queue has been transmitted. This technique is known as "tail drop", since the packet that arrived most recently (i.e., the one on the tail of the queue) is dropped when the queue is full. This method has served the Internet well for years, but it has four important drawbacks:

1. Full Queues

The tail drop discipline allows queues to maintain a full (or, almost full) status for long periods of time, since tail drop signals congestion (via a packet drop) only when the queue has become full. It is important to reduce the steady-state queue size, and this is perhaps the most important goal for queue management.

The naive assumption might be that there is a simple tradeoff between delay and throughput, and that the recommendation that queues be maintained in a "non-full" state essentially translates to a recommendation that low end-to-end delay is more important than high throughput. However, this does not take into account the critical role that packet bursts play in Internet performance. For example, even though TCP constrains the congestion window of a flow, packets often arrive at network devices in bursts [Leland94]. If the queue is full or almost full, an arriving burst will cause multiple packets to be dropped from the same flow. Bursts of loss can result in a global synchronization of flows throttling back, followed by a sustained period of lowered link utilization, reducing overall throughput [Flo94], [Zha90]

The goal of buffering in the network is to absorb data bursts and to transmit them during the (hopefully) ensuing bursts of silence. This is essential to permit transmission of bursts of data. Normally small queues are preferred in network devices, with sufficient queue capacity to absorb the bursts. The counter-intuitive result is that maintaining normally-small queues can result in higher throughput as well as lower end-to-end delay. In summary, queue limits should not reflect the steady state queues we want to be maintained in the network; instead, they should reflect the size of bursts that a network device needs to absorb.

2. Lock-Out

In some situations tail drop allows a single connection or a few flows to monopolize the queue space starving other connections, preventing them from getting room in the queue [Flo92].

3. Mitigating the Impact of Packet Bursts

Large burst of packets can delay other packets, disrupting the control loop (e.g. the pacing of flows by the TCP ACK-Clock), and reducing the performance of flows that share a common bottleneck.

4. Control loop synchronization

Congestion control, like other end-to-end mechanisms, introduces a control loop between hosts. Sessions that share a common network bottleneck can therefore become synchronised, introducing periodic disruption (e.g. jitter/loss). "lock-out" is often also the result of synchronization or other timing effects

Besides tail drop, two alternative queue management disciplines that can be applied when a queue becomes full are "random drop on full" or "head drop on full". When a new packet arrives at a full queue using the random drop on full discipline, the network device drops a randomly selected packet from the queue (which can be an expensive operation, since it naively requires an $O(N)$ walk through the packet queue). When a new packet arrives at a full queue using the head drop on full discipline, the network device drops the packet at the front of the queue [Lakshman96]. Both of these solve the lock-out problem, but neither solves the full-queues problem described above.

We know in general how to solve the full-queues problem for "responsive" flows, i.e., those flows that throttle back in response to congestion notification. In the current Internet, dropped packets provide a critical mechanism indicating congestion notification to hosts. The solution to the full-queues problem is for network devices to drop or ECN-mark packets before a queue becomes full, so that hosts can respond to congestion before buffers overflow. We call such a proactive approach AQM. By dropping or ECN-marking packets before buffers overflow, AQM allows network devices to control when and how many packets to drop.

In summary, an active queue management mechanism can provide the following advantages for responsive flows.

1. Reduce number of packets dropped in network devices

Packet bursts are an unavoidable aspect of packet networks [Willinger95]. If all the queue space in a network device is already committed to "steady state" traffic or if the buffer

space is inadequate, then the network device will have no ability to buffer bursts. By keeping the average queue size small, AQM will provide greater capacity to absorb naturally-occurring bursts without dropping packets.

Furthermore, without AQM, more packets will be dropped when a queue does overflow. This is undesirable for several reasons. First, with a shared queue and the tail drop discipline, this can result in unnecessary global synchronization of flows, resulting in lowered average link utilization, and hence lowered network throughput. Second, unnecessary packet drops represent a waste of network capacity on the path before the drop point.

While AQM can manage queue lengths and reduce end-to-end latency even in the absence of end-to-end congestion control, it will be able to reduce packet drops only in an environment that continues to be dominated by end-to-end congestion control.

2. Provide a lower-delay interactive service

By keeping a small average queue size, AQM will reduce the delays experienced by flows. This is particularly important for interactive applications such as short web transfers, POP/IMAP, DNS, terminal traffic (telnet, ssh, mosh, RDP, etc), gaming or interactive audio-video sessions, whose subjective (and objective) performance is better when the end-to-end delay is low.

3. Avoid lock-out behavior

AQM can prevent lock-out behavior by ensuring that there will almost always be a buffer available for an incoming packet. For the same reason, AQM can prevent a bias against low capacity, but highly bursty, flows.

Lock-out is undesirable because it constitutes a gross unfairness among groups of flows. However, we stop short of calling this benefit "increased fairness", because general fairness among flows requires per-flow state, which is not provided by queue management. For example, in a network device using AQM with only FIFO scheduling, two TCP flows may receive very different share of the network capacity simply because they have different round-trip times [Floyd91], and a flow that does not use congestion control may receive more capacity than a flow that does. AQM can therefore be combined with a scheduling mechanism that divides network traffic between multiple queues (section 2.1).

4. Reduce the probability of control loop synchronization

The probability of network control loop synchronization can be reduced if network devices introduce randomness in the AQM functions that trigger congestion avoidance at the sending host.

2.1. AQM and Multiple Queues

A network device may use per-flow or per-class queuing with a scheduling algorithm to either prioritize certain applications or classes of traffic, limit the rate of transmission, or to provide isolation between different traffic flows within a common class. For example, a router may maintain per-flow state to achieve general fairness by a per-flow scheduling algorithm such as various forms of Fair Queueing (FQ) [Dem90] [Sut99], including Weighted Fair Queueing (WFQ), Stochastic Fairness Queueing (SFQ) [McK90] Deficit Round Robin (DRR) [Shr96], [Nic12], and/or a Class-Based Queue scheduling algorithm such as CBQ [Floyd95]. Hierarchical queues may also be used e.g., as a part of a Hierarchical Token Bucket (HTB), or Hierarchical Fair Service Curve (HFSC) [Sto97]. These methods are also used to realize a range of Quality of Service (QoS) behaviours designed to meet the need of traffic classes (e.g. using the integrated or differentiated service models).

AQM is needed even for network devices that use per-flow or per-class queuing, because scheduling algorithms by themselves do not control the overall queue size or the size of individual queues. AQM mechanisms might need to control the overall queue sizes, to ensure that arriving bursts can be accommodated without dropping packets. AQM should also be used to control the queue size for each individual flow or class, so that they do not experience unnecessarily high delay. Using a combination of AQM and scheduling between multiple queues has been shown to offer good results in experimental and some types of operational use.

In short, scheduling algorithms and queue management should be seen as complementary, not as replacements for each other.

2.2. AQM and Explicit Congestion Marking (ECN)

An AQM method may use Explicit Congestion Notification (ECN) [RFC3168] instead of dropping to mark packets under mild or moderate congestion. ECN-marking can allow a network device to signal congestion at a point before a transport experiences congestion loss or additional queuing delay [ECN-Benefit]. Section 4.2.1 describes some of the benefits of using ECN with AQM.

2.3. AQM and Buffer Size

It is important to differentiate the choice of buffer size for a queue in a switch/router or other network device, and the threshold(s) and other parameters that determine how and when an AQM algorithm operates. The optimum buffer size is a function of operational requirements and should generally be sized to be sufficient to buffer the largest normal traffic burst that is expected. This size depends on the number and burstiness of traffic arriving at the queue and the rate at which traffic leaves the queue.

One objective of AQM is to minimize the effect of lock-out, where one flow prevents other flows from effectively gaining capacity. This need can be illustrated by a simple example of drop-tail queuing when a new TCP flow injects packets into a queue that happens to be almost full. A TCP flow's congestion control algorithm [RFC5681] increases the flow rate to maximize its effective window. This builds a queue in the network, inducing latency to the flow and other flows that share this queue. Once a drop-tail queue fills, there will also be loss. A new flow, sending its initial burst, has an enhanced probability of filling the remaining queue and dropping packets. As a result, the new flow can be effectively prevented from effectively sharing the queue for a period of many RTTs. In contrast, AQM can minimize the mean queue depth and therefore reducing the probability that competing sessions can materially prevent each other from performing well.

AQM frees a designer from having to limit the buffer space assigned to a queue to achieve acceptable performance, allowing allocation of sufficient buffering to satisfy the needs of the particular traffic pattern. Different types of traffic and deployment scenarios will lead to different requirements. The choice of AQM algorithm and associated parameters is therefore a function of the way in which congestion is experienced and the required reaction to achieve acceptable performance. This latter is the primary topic of the following sections.

3. Managing Aggressive Flows

One of the keys to the success of the Internet has been the congestion avoidance mechanisms of TCP. Because TCP "backs off" during congestion, a large number of TCP connections can share a single, congested link in such a way that link bandwidth is shared reasonably equitably among similarly situated flows. The equitable sharing of bandwidth among flows depends on all flows running compatible congestion avoidance algorithms, i.e., methods conformant with the current TCP specification [RFC5681].

In this document a flow is known as "TCP-friendly" when it has a congestion response that approximates the average response expected of a TCP flow. One example method of a TCP-friendly scheme is the TCP-Friendly Rate Control algorithm [RFC5348]. In this document, the term is used more generally to describe this and other algorithms that meet these goals.

There are a variety of types of network flow. Some convenient classes that describe flows are: (1) TCP Friendly flows, (2) unresponsive flows, i.e., flows that do not slow down when congestion occurs, and (3) flows that are responsive but are less responsive to congestion than TCP. The last two classes contain more aggressive flows that can pose significant threats to Internet performance.

1. TCP-Friendly flows

A TCP-friendly flow responds to congestion notification within a small number of path Round Trip Times (RTT), and in steady-state it uses no more capacity than a conformant TCP running under comparable conditions (drop rate, RTT, packet size, etc.). This is described in the remainder of the document.

2. Non-Responsive Flows

A flow that does not adjust its rate in response to congestion notification within a small number of path RTTs, can also use more capacity than a conformant TCP running under comparable conditions. There is a growing set of applications whose congestion avoidance algorithms are inadequate or nonexistent (i.e., a flow that does not throttle its sending rate when it experiences congestion).

The User Datagram Protocol (UDP) [RFC0768] provides a minimal, best-effort transport to applications and upper-layer protocols (both simply called "applications" in the remainder of this document) and does not itself provide mechanisms to prevent congestion collapse and establish a degree of fairness [RFC5405]. Examples that use UDP include some streaming applications for packet voice and video, and some multicast bulk data transport. Other traffic, when aggregated may also become unresponsive to congestion notification. If no action is taken, such unresponsive flows could lead to a new congestion collapse [RFC2914]. Some applications can even increase their traffic volume in response to congestion (e.g. by adding forward error correction when loss is experienced), with the possibility that they contribute to congestion collapse.

In general, applications need to incorporate effective congestion avoidance mechanisms [RFC5405]. Research continues to be needed to identify and develop ways to accomplish congestion avoidance for presently unresponsive applications. Network devices need to be able to protect themselves against unresponsive flows, and mechanisms to accomplish this must be developed and deployed. Deployment of such mechanisms would provide an incentive for all applications to become responsive by either using a congestion-controlled transport (e.g. TCP, SCTP [RFC4960] and DCCP [RFC4340].) or by incorporating their own congestion control in the application [RFC5405], [RFC6679].

3. Transport Flows that are less responsive than TCP

A second threat is posed by transport protocol implementations that are responsive to congestion, but, either deliberately or through faulty implementation, reduce less than a TCP flow would have done in response to congestion. This covers a spectrum of behaviours between (1) and (2). If applications are not sufficiently responsive to congestion signals, they may gain an unfair share of the available network capacity.

For example, the popularity of the Internet has caused a proliferation in the number of TCP implementations. Some of these may fail to implement the TCP congestion avoidance mechanisms correctly because of poor implementation. Others may deliberately be implemented with congestion avoidance algorithms that are more aggressive in their use of capacity than other TCP implementations; this would allow a vendor to claim to have a "faster TCP". The logical consequence of such implementations would be a spiral of increasingly aggressive TCP implementations, leading back to the point where there is effectively no congestion avoidance and the Internet is chronically congested.

Another example could be an RTP/UDP video flow that uses an adaptive codec, but responds incompletely to indications of congestion or responds over an excessively long time period. Such flows are unlikely to be responsive to congestion signals in a timeframe comparable to a small number of end-to-end transmission delays. However, over a longer timescale, perhaps seconds in duration, they could moderate their speed, or increase their speed if they determine capacity to be available.

Tunneled traffic aggregates carrying multiple (short) TCP flows can be more aggressive than standard bulk TCP. Applications (e.g., web browsers primarily supporting HTTP 1.1 and peer-to-peer file-sharing) have exploited this by opening multiple connections to the same endpoint.

Lastly, some applications (e.g., web browsers primarily supporting HTTP 1.1) open a large numbers of successive short TCP flows for a single session. This can lead to each individual flow spending the majority of time in the exponential TCP slow start phase, rather than in TCP congestion avoidance. The resulting traffic aggregate can therefore be much less responsive than a single standard TCP flow.

The projected increase in the fraction of total Internet traffic for more aggressive flows in classes 2 and 3 could pose a threat to the performance of the future Internet. There is therefore an urgent need for measurements of current conditions and for further research into the ways of managing such flows. This raises many difficult issues in finding methods with an acceptable overhead cost that can identify and isolate unresponsive flows or flows that are less responsive than TCP. Finally, there is as yet little measurement or simulation evidence available about the rate at which these threats are likely to be realized, or about the expected benefit of algorithms for managing such flows.

Another topic requiring consideration is the appropriate granularity of a "flow" when considering a queue management method. There are a few "natural" answers: 1) a transport (e.g., TCP or UDP) flow (source address/port, destination address/port, protocol); 2) Differentiated Services Code Point, DSCP; 3) a source/destination host pair (IP address); 4) a given source host or a given destination host, or various combinations of the above; 5) a subscriber or site receiving the Internet service (enterprise or residential).

The source/destination host pair gives an appropriate granularity in many circumstances. However, different vendors/providers use different granularities for defining a flow (as a way of "distinguishing" themselves from one another), and different granularities may be chosen for different places in the network. It may be the case that the granularity is less important than the fact that a network device needs to be able to deal with more unresponsive flows at *some* granularity. The granularity of flows for congestion management is, at least in part, a question of policy that needs to be addressed in the wider IETF community.

4. Conclusions and Recommendations

The IRTF, in publishing [RFC2309], and the IETF in subsequent discussion, has developed a set of specific recommendations regarding the implementation and operational use of AQM procedures. The recommendations provided by this document are summarised as:

1. Network devices SHOULD implement some AQM mechanism to manage queue lengths, reduce end-to-end latency, and avoid lock-out phenomena within the Internet.
2. Deployed AQM algorithms SHOULD support Explicit Congestion Notification (ECN) as well as loss to signal congestion to endpoints.
3. AQM algorithms SHOULD NOT require tuning of initial or configuration parameters in common use cases.
4. AQM algorithms SHOULD respond to measured congestion, not application profiles.
5. AQM algorithms SHOULD NOT interpret specific transport protocol behaviours.
6. Transport protocol congestion control algorithms SHOULD maximize their use of available capacity (when there is data to send) without incurring undue loss or undue round trip delay.
7. Research, engineering, and measurement efforts are needed regarding the design of mechanisms to deal with flows that are unresponsive to congestion notification or are responsive, but are more aggressive than present TCP.

These recommendations are expressed using the word "SHOULD". This is in recognition that there may be use cases that have not been envisaged in this document in which the recommendation does not apply. Therefore, care should be taken in concluding that one's use case falls in that category; during the life of the Internet, such use cases have been rarely if ever observed and reported. To the contrary, available research [Choi04] says that even high speed links in network cores that are normally very stable in depth and behavior experience occasional issues that need moderation. The recommendations are detailed in the following sections.

4.1. Operational deployments SHOULD use AQM procedures

AQM procedures are designed to minimize the delay and buffer exhaustion induced in the network by queues that have filled as a result of host behavior. Marking and loss behaviors provide a signal that buffers within network devices are becoming unnecessarily full, and that the sender would do well to moderate its behavior.

The use of scheduling mechanisms, such as priority queuing, classful queuing, and fair queuing, is often effective in networks to help a network serve the needs of a range of applications. Network

operators can use these methods to manage traffic passing a choke point. This is discussed in [RFC2474] and [RFC2475]. When scheduling is used AQM should be applied across the classes or flows as well as within each class or flow:

- o AQM mechanisms need to control the overall queue sizes, to ensure that arriving bursts can be accommodated without dropping packets.
- o AQM mechanisms need to allow combination with other mechanisms, such as scheduling, to allow implementation of policies for providing fairness between different flows.
- o AQM should be used to control the queue size for each individual flow or class, so that they do not experience unnecessarily high delay.

4.2. Signaling to the transport endpoints

There are a number of ways a network device may signal to the end point that the network is becoming congested and trigger a reduction in rate. The signalling methods include:

- o Delaying transport segments (packets) in flight, such as in a queue.
- o Dropping transport segments (packets) in transit.
- o Marking transport segments (packets), such as using Explicit Congestion Control [RFC3168] [RFC4301] [RFC4774] [RFC6040] [RFC6679].

Increased network latency is used as an implicit signal of congestion. E.g., in TCP additional delay can affect ACK Clocking and has the result of reducing the rate of transmission of new data. In the Real Time Protocol (RTP), network latency impacts the RTCP-reported RTT and increased latency can trigger a sender to adjust its rate. Methods such as Low Extra Delay Background Transport (LEDBAT) [RFC6817] assume increased latency as a primary signal of congestion. Appropriate use of delay-based methods and the implications of AQM presently remains an area for further research.

It is essential that all Internet hosts respond to loss [RFC5681], [RFC5405] [RFC4960] [RFC4340]. Packet dropping by network devices that are under load has two effects: It protects the network, which is the primary reason that network devices drop packets. The detection of loss also provides a signal to a reliable transport (e.g., TCP, SCTP) that there is potential congestion using a pragmatic heuristic; "when the network discards a message in flight, it may imply the presence

of faulty equipment or media in a path, and it may imply the presence of congestion. To be conservative, a transport must assume it may be the latter." Applications using unreliable transports (e.g., using UDP) need to similarly react to loss [RFC5405]

Network devices SHOULD use an AQM algorithm to measure local congestion and to determine the packets to mark or drop so that the congestion is managed.

In general, dropping multiple packets from the same sessions in the same RTT is ineffective, and can reduce throughput. Also, dropping or marking packets from multiple sessions simultaneously can have the effect of synchronizing them, resulting in increasing peaks and troughs in the subsequent traffic load. Hence, AQM algorithms SHOULD randomize dropping in time, to reduce the probability that congestion indications are only experienced by a small proportion of the active flows.

Loss due to dropping also has an effect on the efficiency of a flow and can significantly impact some classes of application. In reliable transports the dropped data must be subsequently retransmitted. While other applications/transport may adapt to the absence of lost data, this still implies inefficient use of available capacity and the dropped traffic can affect other flows. Hence, congestion signalling by loss is not entirely positive; it is a necessary evil.

4.2.1. AQM and ECN

Explicit Congestion Notification (ECN) [RFC4301] [RFC4774] [RFC6040] [RFC6679] is a network-layer function that allows a transport to receive network congestion information from a network device without incurring the unintended consequences of loss. ECN includes both transport mechanisms and functions implemented in network devices, the latter rely upon using AQM to decide when and whether to ECN-mark.

Congestion for ECN-capable transports is signalled by a network device setting the "Congestion Experienced (CE)" codepoint in the IP header. This codepoint is noted by the remote receiving end point and signalled back to the sender using a transport protocol mechanism, allowing the sender to trigger timely congestion control. The decision to set the CE codepoint requires an AQM algorithm configured with a threshold. Non-ECN capable flows (the default) are dropped under congestion.

Network devices SHOULD use an AQM algorithm that marks ECN-capable traffic when making decisions about the response to congestion.

Network devices need to implement this method by marking ECN-capable traffic or by dropping non-ECN-capable traffic.

Safe deployment of ECN requires that network devices drop excessive traffic, even when marked as originating from an ECN-capable transport. This is a necessary safety precaution because:

1. A non-conformant, broken or malicious receiver could conceal an ECN mark, and not report this to the sender;
2. A non-conformant, broken or malicious sender could ignore a reported ECN mark, as it could ignore a loss without using ECN;
3. A malfunctioning or non-conforming network device may "hide" an ECN mark (or fail to correctly set the ECN codepoint at an egress of a network tunnel).

In normal operation, such cases should be very uncommon, however overload protection is desirable to protect traffic from misconfigured or malicious use of ECN (e.g., a denial-of-service attack that generates ECN-capable traffic that is unresponsive to CE-marking).

An AQM algorithm that supports ECN needs to define the threshold and algorithm for ECN-marking. This threshold MAY differ from that used for dropping packets that are not marked as ECN-capable, and SHOULD be configurable.

Network devices SHOULD use an algorithm to drop excessive traffic (e.g., at some level above the threshold for CE-marking), even when the packets are marked as originating from an ECN-capable transport.

4.3. AQM algorithm deployment SHOULD NOT require operational tuning

A number of AQM algorithms have been proposed. Many require some form of tuning or setting of parameters for initial network conditions. This can make these algorithms difficult to use in operational networks.

AQM algorithms need to consider both "initial conditions" and "operational conditions". The former includes values that exist before any experience is gathered about the use of the algorithm, such as the configured speed of interface, support for full duplex communication, interface MTU and other properties of the link. The latter includes information observed from monitoring the size of the queue, experienced queueing delay, rate of packet discard, etc.

This document therefore specifies that AQM algorithms that are proposed for deployment in the Internet have the following properties:

- o AQM algorithm deployment SHOULD NOT require tuning. An algorithm MUST provide a default behaviour that auto-tunes to a reasonable performance for typical network operational conditions. This is expected to ease deployment and operation. Initial conditions, such as the interface rate and MTU size or other values derived from these, MAY be required by an AQM algorithm.
- o MAY support further manual tuning that could improve performance in a specific deployed network. Algorithms that lack such variables are acceptable, but if such variables exist, they SHOULD be externalized (made visible to the operator). Guidance needs to be provided on the cases where auto-tuning is unlikely to achieve acceptable performance and to identify the set of parameters that can be tuned. For example, the expected response of an algorithm may need to be configured to accommodate the largest expected Path RTT, since this value can not be known at initialization. This guidance is expected to enable the algorithm to be deployed in networks that have specific characteristics (paths with variable/larger delay; networks where capacity is impacted by interactions with lower layer mechanisms, etc).
- o MAY provide logging and alarm signals to assist in identifying if an algorithm using manual or auto-tuning is functioning as expected. (e.g., this could be based on an internal consistency check between input, output, and mark/drop rates over time). This is expected to encourage deployment by default and allow operators to identify potential interactions with other network functions.

Hence, self-tuning algorithms are to be preferred. Algorithms recommended for general Internet deployment by the IETF need to be designed so that they do not require operational (especially manual) configuration or tuning.

4.4. AQM algorithms SHOULD respond to measured congestion, not application profiles.

Not all applications transmit packets of the same size. Although applications may be characterized by particular profiles of packet size this should not be used as the basis for AQM (see next section). Other methods exist, e.g., Differentiated Services queueing, Pre-Congestion Notification (PCN) [RFC5559], that can be used to differentiate and police classes of application. Network devices may combine AQM with these traffic classification mechanisms and perform AQM only on specific queues within a network device.

An AQM algorithm should not deliberately try to prejudice the size of packet that performs best (i.e., Preferentially drop/mark based only on packet size). Procedures for selecting packets to mark/drop SHOULD observe the actual or projected time that a packet is in a queue (bytes at a rate being an analog to time). When an AQM algorithm decides whether to drop (or mark) a packet, it is RECOMMENDED that the size of the particular packet should not be taken into account [RFC7141].

Applications (or transports) generally know the packet size that they are using and can hence make their judgments about whether to use small or large packets based on the data they wish to send and the expected impact on the delay or throughput, or other performance parameter. When a transport or application responds to a dropped or marked packet, the size of the rate reduction should be proportionate to the size of the packet that was sent [RFC7141].

AQM-enabled system MAY instantiate different instances of an AQM algorithm to be applied within the same traffic class. Traffic classes may be differentiated based on an Access Control List (ACL), the packet Differentiated Services Code Point (DSCP) [RFC5559], enabling use of the ECN field (i.e., any of ECT(0), ECT(1) or CE) [RFC3168] [RFC4774], a multi-field (MF) classifier that combines the values of a set of protocol fields (e.g., IP address, transport, ports) or an equivalent codepoint at a lower layer. This recommendation goes beyond what is defined in RFC 3168, by allowing that an implementation MAY use more than one instance of an AQM algorithm to handle both ECN-capable and non-ECN-capable packets.

4.5. AQM algorithms SHOULD NOT be dependent on specific transport protocol behaviours

In deploying AQM, network devices need to support a range of Internet traffic and SHOULD NOT make implicit assumptions about the characteristics desired by the set transports/applications the network supports. That is, AQM methods should be opaque to the choice of transport and application.

AQM algorithms are often evaluated by considering TCP [RFC0793] with a limited number of applications. Although TCP is the predominant transport in the Internet today, this no longer represents a sufficient selection of traffic for verification. There is significant use of UDP [RFC0768] in voice and video services, and some applications find utility in SCTP [RFC4960] and DCCP [RFC4340]. Hence, AQM algorithms should also demonstrate operation with transports other than TCP and need to consider a variety of applications. Selection of AQM algorithms also needs to consider use of tunnel encapsulations that may carry traffic aggregates.

AQM algorithms SHOULD NOT target or derive implicit assumptions about the characteristics desired by specific transports/applications. Transports and applications need to respond to the congestion signals provided by AQM (i.e., dropping or ECN-marking) in a timely manner (within a few RTT at the latest).

4.6. Interactions with congestion control algorithms

Applications and transports need to react to received implicit or explicit signals that indicate the presence of congestion. This section identifies issues that can impact the design of transport protocols when using paths that use AQM.

Transport protocols and applications need timely signals of congestion. The time taken to detect and respond to congestion is increased when network devices queue packets in buffers. It can be difficult to detect tail losses at a higher layer and this may sometimes require transport timers or probe packets to detect and respond to such loss. Loss patterns may also impact timely detection, e.g., the time may be reduced when network devices do not drop long runs of packets from the same flow.

A common objective of an elastic transport congestion control protocol is to allow an application to deliver the maximum rate of data without inducing excessive delays when packets are queued in a buffers within the network. To achieve this, a transport should try to operate at rate below the inflexion point of the load/delay curve (the bend of what is sometimes called a "hockey-stick" curve) [Jain94]. When the congestion window allows the load to approach this bend, the end-to-end delay starts to rise - a result of congestion, as packets probabilistically arrive at non-overlapping times. On the one hand, a transport that operates above this point can experience congestion loss and could also trigger operator activities, such as those discussed in [RFC6057]. On the other hand, a flow may achieve both near-maximum throughput and low latency when it operates close to this knee point, with minimal contribution to router congestion. Choice of an appropriate rate/congestion window can therefore significantly impact the loss and delay experienced by a flow and will impact other flows that share a common network queue.

Some applications may send less than permitted by the congestion control window (or rate). Examples include multimedia codecs that stream at some natural rate (or set of rates) or an application that is naturally interactive (e.g., some web applications, interactive server-based gaming, transaction-based protocols). Such applications may have different objectives. They may not wish to maximize throughput, but may desire a lower loss rate or bounded delay.

The correct operation of an AQM-enabled network device MUST NOT rely upon specific transport responses to congestion signals.

4.7. The need for further research

The second recommendation of [RFC2309] called for further research into the interaction between network queues and host applications, and the means of signaling between them. This research has occurred, and we as a community have learned a lot. However, we are not done.

We have learned that the problems of congestion, latency and buffer-sizing have not gone away, and are becoming more important to many users. A number of self-tuning AQM algorithms have been found that offer significant advantages for deployed networks. There is also renewed interest in deploying AQM and the potential of ECN.

Traffic patterns can depend on the network deployment scenario, and Internet research therefore needs to consider the implications of a diverse range of application interactions. This includes ensuring that combinations of mechanisms, as well as combinations of traffic patterns, do not interact and result in either significantly reduced flow throughput or significantly increased latency.

At the time of writing (in 2015), an obvious example of further research is the need to consider the many-to-one communication patterns found in data centers, known as incast [Ren12], (e.g., produced by Map/Reduce applications). Such analysis needs to study not only each application traffic type, but should also include combinations of types of traffic.

Research also needs to consider the need to extend our taxonomy of transport sessions to include not only "mice" and "elephants", but "lemmings"? Where "Lemmings" are flash crowds of "mice" that the network inadvertently tries to signal to as if they were elephant flows, resulting in head of line blocking in a data center deployment scenario.

Examples of other required research include:

- o Research into new AQM and scheduling algorithms.
- o Appropriate use of delay-based methods and the implications of AQM.
- o Research into suitable algorithms for marking ECN-capable packets that do not require operational configuration or tuning for common use.

- o Experience in the deployment of ECN alongside AQM.
- o Tools for enabling AQM (and ECN) deployment and measuring the performance.
- o Methods for mitigating the impact of non-conformant and malicious flows.
- o Research to understand the implications of using new network and transport methods on applications.

Hence, this document therefore reiterates the call of RFC 2309: we need continuing research as applications develop.

5. IANA Considerations

This memo asks the IANA for no new parameters.

6. Security Considerations

While security is a very important issue, it is largely orthogonal to the performance issues discussed in this memo.

This recommendation requires algorithms to be independent of specific transport or application behaviors. Therefore a network device does not require visibility or access to upper layer protocol information to implement an AQM algorithm. This ability to operate in an application-agnostic fashion is therefore an example of a privacy-enhancing feature.

Many deployed network devices use queueing methods that allow unresponsive traffic to capture network capacity, denying access to other traffic flows. This could potentially be used as a denial-of-service attack. This threat could be reduced in network devices that deploy AQM or some form of scheduling. We note, however, that a denial-of-service attack that results in unresponsive traffic flows may be indistinguishable from other traffic flows (e.g., tunnels carrying aggregates of short flows, high-rate isochronous applications). New methods therefore may remain vulnerable, and this document recommends that ongoing research should consider ways to mitigate such attacks.

7. Privacy Considerations

This document, by itself, presents no new privacy issues.

8. Acknowledgements

The original version of this document describing best current practice was based on the informational text of [RFC2309]. This was written by the End-to-End Research Group, which is to say Bob Braden, Dave Clark, Jon Crowcroft, Bruce Davie, Steve Deering, Deborah Estrin, Sally Floyd, Van Jacobson, Greg Minshall, Craig Partridge, Larry Peterson, KK Ramakrishnan, Scott Shenker, John Wroclawski, and Lixia Zhang. Although there are important differences, many of the key arguments in the present document remain unchanged from those in RFC 2309.

The need for an updated document was agreed to in the tsvarea meeting at IETF 86. This document was reviewed on the aqm@ietf.org list. Comments were received from Colin Perkins, Richard Scheffenegger, Dave Taht, John Leslie, David Collier-Brown and many others.

Gorry Fairhurst was in part supported by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700).

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.
- [RFC4774] Floyd, S., "Specifying Alternate Semantics for the Explicit Congestion Notification (ECN) Field", BCP 124, RFC 4774, November 2006.
- [RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, November 2008.
- [RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.
- [RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, November 2010.

- [RFC6679] Westerlund, M., Johansson, I., Perkins, C., O'Hanlon, P., and K. Carlberg, "Explicit Congestion Notification (ECN) for RTP over UDP", RFC 6679, August 2012.
- [RFC7141] Briscoe, B. and J. Manner, "Byte and Packet Congestion Notification", BCP 41, RFC 7141, February 2014.

9.2. Informative References

- [AQM-WG] "IETF AQM WG", .
- [Bri15] Briscoe, Bob., Brunstrom, Anna., Petlund, Andreas., Hayes, David., Ros, David., Tsang, Ing-Jyh., Gjessing, Stein., Fairhurst, Gorrry., Griwodz, Carsten., and Michael. Welzl, "Reducing Internet Latency: A Survey of Techniques and their Merit, IEEE Communications Surveys & Tutorials", 2015.
- [CONEX] Mathis, M. and B. Briscoe, "The Benefits to Applications of using Explicit Congestion Notification (ECN)", IETF (Work-in-Progress) draft-ietf-conex-abstract-mech, March 2014.
- [Choi04] Choi, Baek-Young., Moon, Sue., Zhang, Zhi-Li., Papagiannaki, K., and C. Diot, "Analysis of Point-To-Point Packet Delay In an Operational Network", March 2004.
- [Dem90] Demers, A., Keshav, S., and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm, Internetworking: Research and Experience", SIGCOMM Symposium proceedings on Communications architectures and protocols , 1990.
- [ECN-Benefit] Welzl, M. and G. Fairhurst, "The Benefits to Applications of using Explicit Congestion Notification (ECN)", IETF (Work-in-Progress) , February 2014.
- [Flo92] Floyd, S. and V. Jacobsen, "On Traffic Phase Effects in Packet-Switched Gateways", 1992.
- [Flo94] Floyd, S. and V. Jacobsen, "The Synchronization of Periodic Routing Messages, http://ee.lbl.gov/papers/sync_94.pdf", 1994.
- [Floyd91] Floyd, S., "Connections with Multiple Congested Gateways in Packet-Switched Networks Part 1: One-way Traffic.", Computer Communications Review , October 1991.

- [Floyd95] Floyd, S. and V. Jacobson, "Link-sharing and Resource Management Models for Packet Networks", IEEE/ACM Transactions on Networking , August 1995.
- [Jacobson88] Jacobson, V., "Congestion Avoidance and Control", SIGCOMM Symposium proceedings on Communications architectures and protocols , August 1988.
- [Jain94] Jain, Raj., Ramakrishnan, KK., and Chiu. Dah-Ming, "Congestion avoidance scheme for computer networks", US Patent Office 5377327, December 1994.
- [Lakshman96] Lakshman, TV., Neidhardt, A., and T. Ott, "The Drop From Front Strategy in TCP Over ATM and Its Interworking with Other Control Features", IEEE Infocomm , 1996.
- [Leland94] Leland, W., Taqqu, M., Willinger, W., and D. Wilson, "On the Self-Similar Nature of Ethernet Traffic (Extended Version)", IEEE/ACM Transactions on Networking , February 1994.
- [McK90] McKenney, PE. and G. Varghese, "Stochastic Fairness Queuing", <http://www2.rdrop.com/~paulmck/scalability/paper/sfq.2002.06.04.pdf> , 1990.
- [Nic12] Nichols, K., "Controlling Queue Delay", Communications of the ACM Vol. 55 No. 11, July, 2012, pp.42-50. , July 2002.
- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.
- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.
- [RFC0896] Nagle, J., "Congestion control in IP/TCP internetworks", RFC 896, January 1984.
- [RFC0970] Nagle, J., "On packet switches with infinite storage", RFC 970, December 1985.

- [RFC1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.
- [RFC1633] Braden, B., Clark, D., and S. Shenker, "Integrated Services in the Internet Architecture: an Overview", RFC 1633, June 1994.
- [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, April 1998.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [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, December 1998.
- [RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.
- [RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, September 2000.
- [RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, March 2006.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [RFC5348] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 5348, September 2008.
- [RFC5559] Eardley, P., "Pre-Congestion Notification (PCN) Architecture", RFC 5559, June 2009.
- [RFC6057] Bastian, C., Klieber, T., Livingood, J., Mills, J., and R. Woundy, "Comcast's Protocol-Agnostic Congestion Management System", RFC 6057, December 2010.

- [RFC6789] Briscoe, B., Woundy, R., and A. Cooper, "Congestion Exposure (ConEx) Concepts and Use Cases", RFC 6789, December 2012.
- [RFC6817] Shalunov, S., Hazel, G., Iyengar, J., and M. Kuehlewind, "Low Extra Delay Background Transport (LEDBAT)", RFC 6817, December 2012.
- [RFC7414] Duke, M., Braden, R., Eddy, W., Blanton, E., and A. Zimmermann, "A Roadmap for Transmission Control Protocol (TCP) Specification Documents", RFC 7414, February 2015.
- [Ren12] Ren, Y., Zhao, Y., and P. Liu, "A survey on TCP Incast in data center networks, International Journal of Communication Systems, Volume 27, Issue 8, pages 1160-117", 1990.
- [Shr96] Shreedhar, M. and G. Varghese, "Efficient Fair Queueing Using Deficit Round Robin", IEEE/ACM Transactions on Networking Vol 4, No. 3 , July 1996.
- [Sto97] Stoica, I. and H. Zhang, "A Hierarchical Fair Service Curve algorithm for Link sharing, real-time and priority services", ACM SIGCOMM , 1997.
- [Sut99] Suter, B., "Buffer Management Schemes for Supporting TCP in Gigabit Routers with Per-flow Queueing", IEEE Journal on Selected Areas in Communications Vol. 17 Issue 6, June, 1999, pp. 1159-1169. , 1999.
- [Willinger95] Willinger, W., Taqqu, M., Sherman, R., Wilson, D., and V. Jacobson, "Self-Similarity Through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level", SIGCOMM Symposium proceedings on Communications architectures and protocols , August 1995.
- [Zha90] Zhang, L. and D. Clark, "Oscillating Behavior of Network Traffic: A Case Study Simulation, <http://groups.csail.mit.edu/ana/Publications/Zhang-DDC-Oscillating-Behavior-of-Network-Traffic-1990.pdf>", 1990.

Appendix A. Change Log

RFC-Editor please remove this appendix before publication.

Initial Version: March 2013

Minor update of the algorithms that the IETF recommends SHOULD NOT require operational (especially manual) configuration or tuning
April 2013

Major surgery. This draft is for discussion at IETF-87 and expected to be further updated.
July 2013

-00 WG Draft - Updated transport recommendations; revised deployment configuration section; numerous minor edits.
Oct 2013

-01 WG Draft - Updated transport recommendations; revised deployment configuration section; numerous minor edits.
Jan 2014 - Feedback from WG.

-02 WG Draft - Minor edits Feb 2014 - Mainly language fixes.

-03 WG Draft - Minor edits Feb 2013 - Comments from David Collier-Brown and David Taht.

-04 WG Draft - Minor edits May 2014 - Comments during WGLC: Provided some introductory subsections to help people (with subsections and better text). - Written more on the role scheduling. - Clarified that ECN mark threshold needs to be configurable. - Reworked your "knee" para. Various updates in response to feedback.

-05 WG Draft - Minor edits June 2014 - New text added to address further comments, and improve introduction - adding context, reference to Conex, linking between sections, added text on synchronization.

-06 WG Draft - Minor edits July 2014 - Reorganised the introduction following WG feedback to better explain how this relates to the original goals of RFC2309. Added item on packet bursts. Various minor corrections incorporated - no change to main recommendations.

-07 WG Draft - Minor edits July 2014 - Replaced ID REF by RFC 7141. Changes made to introduction following inputs from Wes Eddy and John Leslie. Corrections and additions proposed by Bob Briscoe.

-08 WG Draft - Minor edits August 2014 - Review comments from John Leslie and Bob Briscoe. Text corrections including; updated Acknowledgments (RFC2309 ref) s/congestive/congestion/g; changed the more bold language from RFC2309 to reflect a more considered perceived threat to Internet Performance; modified the category that is not-TCP-like to be "less responsive to congestion than

TCP" and more clearly noted that represents a range of behaviours.

-09 WG Draft - Minor edits Jan 2015 - Edits following LC comments.

-10 WG Draft - Minor edits Feb 2015 - Update following IESG Review

-11 WG Draft - Minor edits Feb 2015 - Resolution of last issues.

Authors' Addresses

Fred Baker (editor)
Cisco Systems
Santa Barbara, California 93117
USA

Email: fred@cisco.com

Godred Fairhurst (editor)
University of Aberdeen
School of Engineering
Fraser Noble Building
Aberdeen, Scotland AB24 3UE
UK

Email: gorry@erg.abdn.ac.uk
URI: <http://www.erg.abdn.ac.uk>

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: February 10, 2015

N. Kuhn, Ed.
Telecom Bretagne
P. Natarajan, Ed.
Cisco Systems
D. Ros
Simula Research Laboratory AS
N. Khademi
University of Oslo
August 11, 2014

AQM Characterization Guidelines
draft-kuhn-aqm-eval-guidelines-02

Abstract

Unmanaged large buffers in today's networks have given rise to a slew of performance issues. These performance issues can be addressed by some form of Active Queue Management (AQM), optionally in combination with a packet scheduling scheme such as fair queuing. The IETF AQM and packet scheduling working group was formed to standardize AQM schemes that are robust, easily implemented, and successfully deployed in today's networks. This document describes various criteria for performing precautionary characterizations of AQM proposals. This document also helps in ascertaining whether any given AQM proposal should be taken up for standardization by the AQM WG.

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 <http://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 February 10, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Guidelines for AQM designers	4
1.2. Reducing the latency and maximizing the goodput	5
1.3. Glossary	5
1.4. Requirements Language	5
2. End-to-end metrics	5
2.1. Flow Completion time	6
2.2. Packet loss	6
2.3. Packet loss synchronization	6
2.4. Goodput	6
2.5. Latency and jitter	7
2.6. Discussion on the trade-off between latency and goodput	7
3. Generic set up for evaluations	7
3.1. Topology and notations	8
3.2. Buffer size	9
3.3. Congestion controls	9
4. Various TCP variants	9
4.1. TCP-friendly Sender	10
4.2. Aggressive Transport Sender	10
4.3. Unresponsive Transport Sender	10
4.4. TCP initial congestion window	10
4.5. Traffic Mix	11
5. RTT fairness	12
5.1. Motivation	12
5.2. Required tests	12
5.3. Metrics to evaluate the RTT fairness	13
6. Burst absorption	13
6.1. Motivation	13
6.2. Required tests	14
6.3. Metrics to evaluate the burst absorption capacity	14
7. Stability	14
7.1. Motivation	14
7.2. Required tests	15
7.2.1. Mild Congestion	15
7.2.2. Medium Congestion	15
7.2.3. Heavy Congestion	15
7.2.4. Varying Available Bandwidth	15
7.3. Parameter sensitivity and stability analysis	16
8. Implementation cost	16
8.1. Motivation	16
8.2. Required discussion	17
9. Operator control knobs and auto-tuning	17

10. Interaction with ECN	18
10.1. Motivation	18
10.2. Required discussion	18
11. Interaction with scheduling	18
11.1. Motivation	18
11.2. Required discussion	18
12. Discussion on methodology, metrics, AQM comparisons and packet sizes	18
12.1. Methodology	18
12.2. Comments on metrics measurement	19
12.3. Comparing AQM schemes	19
12.3.1. Performance comparison	19
12.3.2. Deployment comparison	20
12.4. Packet sizes and congestion notification	20
13. Acknowledgements	20
14. Contributors	20
15. IANA Considerations	21
16. Security Considerations	21
17. References	21
17.1. Normative References	21
17.2. Informative References	21
Authors' Addresses	22

1. Introduction

Active Queue Management (AQM) addresses the concerns arising from using unnecessarily large and unmanaged buffers in order to improve network and application performance. Several AQM algorithms have been proposed in the past years, most notable being Random Early Detection (RED), BLUE, and Proportional Integral controller (PI), and more recently CoDel [CODEL] and PIE [PIE]. In general, these algorithms actively interact with the Transmission Control Protocol (TCP) and any other transport protocol that deploys a congestion control scheme to manage the amount of data they keep in the network. The available buffer space in the routers and switches is large enough to accommodate the short-term buffering requirements. AQM schemes aim at reducing mean buffer occupancy, and therefore both end-to-end delay and jitter. Some of these algorithms, notably RED, have also been widely implemented in some network devices. However, any potential benefits of the RED AQM scheme have not been realized since RED is reported to be usually turned off. The main reason of this reluctance to use RED in today's deployments is its sensitivity to the operating conditions in the network and the difficulty of tuning its parameters.

A buffer is a physical volume of memory in which a queue or set of queues are stored. In real implementations of switches, a global memory is shared between the available devices: the size of the buffer for a given communication does not make sense, as its dedicated memory may vary over the time and real world buffering architectures are complex. For the sake of simplicity, when speaking of a specific queue in this document, "buffer size" refers to the

maximum amount of data the buffer may store, which may be measured in bytes or packets. The rest of this memo therefore refers to the maximum queue depth as the size of the buffer for a given communication.

In order to meet mostly throughput-based SLA requirements and to avoid packet drops, many home gateway manufacturers resort to increasing the available memory beyond "reasonable values". This increase is also referred to as Bufferbloat [BB2011]. Deploying large unmanaged buffers on the Internet has led to the increase in end-to-end delay, resulting in poor performance for latency sensitive applications such as real-time multimedia (e.g., voice, video, gaming, etc.). The degree to which this affects modern networking equipment, especially consumer-grade equipment, produces problems even with commonly used web services. Active queue management is thus essential to control queuing delay and decrease network latency.

The AQM and Packet Scheduling working group was recently formed within the TSV area to address the problems with large unmanaged buffers in the Internet. Specifically, the AQM WG is tasked with standardizing AQM schemes that not only address concerns with such buffers, but also that are robust under a wide variety of operating conditions. In order to ascertain whether the WG should undertake standardizing an AQM proposal, the WG requires guidelines for assessing AQM proposals. This document provides the necessary characterization guidelines.

1.1. Guidelines for AQM designers

One of the key objectives behind formulating the guidelines is to help ascertain whether a specific AQM is not only better than drop-tail but also safe to deploy. The guidelines help to quantify AQM schemes' performance in terms of latency reduction, goodput maximization and the trade-off between the two. The guidelines also help to discuss AQM's safe deployment, including self adaptation, stability analysis, fairness, design/implementation complexity and robustness to different operating conditions.

This memo details generic characterization scenarios that any AQM proposal MUST be evaluated against. Irrespective of whether or not an AQM is standardized by the WG, we recommend the relevant scenarios and metrics discussed in this document to be considered. This document presents central aspects of an AQM algorithm that MUST be considered whatever the context is, such as burst absorption capacity, RTT fairness or resilience to fluctuating network conditions. These guidelines could not cover every possible aspect of a particular algorithm. In addition, it is worth noting that the proposed criteria are not bound to a particular evaluation toolset. These guidelines do not present context dependent scenarios (such as

Wi-Fi, data-centers or rural broadband).

This document details how an AQM designer can rate the feasibility of their proposal in different types of network devices (switches, routers, firewalls, hosts, drivers, etc.) where an AQM may be implemented.

1.2. Reducing the latency and maximizing the goodput

The trade-off between reducing the latency and maximizing the goodput is intrinsically linked to each AQM scheme and is key to evaluating its performance. This trade-off MUST be considered in various scenarios to ensure the safety of an AQM deployment. Whenever possible, solutions should aim at both maximizing goodput and minimizing latency. This document proposes guidelines that enable the reader to quantify (1) reduction of latency, (2) maximization of goodput and (3) the trade-off between the two.

Testers SHOULD discuss in a reference document the performance of their proposal in terms of performance and deployment in regards with those of drop-tail: basically, these guidelines provide the tools to understand the deployment costs versus the potential gain in performance of the introduction of the proposed scheme.

1.3. Glossary

- o AQM: there may be confusion whether a scheduling scheme is added to an AQM or is a part of the AQM. The rest of this memo refers to AQM as a dropping policy that does not feature a scheduling scheme.
- o buffer: a physical volume of memory in which a queue or set of queues are stored.
- o buffer size: the maximum amount of data that may be stored in a buffer, measured in bytes or packets.

1.4. 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].

2. End-to-end metrics

End-to-end delay is the result of propagation delay, serialization delay, service delay in a switch, medium-access delay and queuing delay, summed over the network elements in the path. AQM algorithms may reduce the queuing delay by providing signals to the sender on the emergence of congestion, but any impact on the goodput must be carefully considered. This section presents the metrics that SHOULD be used to better quantify (1) the reduction of latency, (2) maximization of goodput and (3) the trade-off between the two. These metrics SHOULD be considered to better assess the performance of an AQM scheme.

2.1. Flow Completion time

The flow completion time is an important performance metric for the end user. Considering the fact that an AQM scheme may drop packets, the flow completion time is directly linked to the dropping policy of the AQM scheme. This metric helps to better assess the performance of an AQM depending on the flow size.

2.2. Packet loss

Packet losses, that may occur in a queue, impact on the end-to-end performance at the receiver's side.

The tester MUST evaluate, at the receiver:

- o the long term packet loss probability;
- o the interval between consecutive losses;
- o the packet loss pattern.

2.3. Packet loss synchronization

One goal of an AQM algorithm should be to help with avoiding global synchronization of flows going through the bottleneck buffer on which the AQM operates ([RFC2309]). It is therefore important to assess the "degree" of packet-loss synchronization between flows, with and without the AQM under consideration.

As discussed e.g. in [LOSS-SYNCH-MET-08], loss synchronization among flows may be quantified by several, slightly different, metrics that capture different aspects of the same issue. However, in real-world measurements the choice of metric may be imposed by practical considerations (e.g., is there fine-grained information on packet losses in the bottleneck available or not). For the purpose of AQM characterization, a good candidate metric is the global synchronization ratio, measuring the proportion of flows losing packets during a loss event. [YU06] used this metric in real-world experiments to characterize synchronization along arbitrary Internet paths; the full methodology is described in [YU06].

2.4. Goodput

Measuring the goodput enables an end-to-end appreciation of how well the AQM improves transport and application performance. The measured end-to-end goodput is linked to the AQM scheme's dropping policy -- the smaller the packet drops, the fewer packets need retransmission, minimizing AQM's impact on transport and application performance. Additionally, an AQM scheme may resort to Explicit Congestion Notification (ECN) marking as an initial means to control delay. Again, marking packets instead of dropping them reduces number of packet retransmissions and increases goodput. Overall, end-to-end goodput values help evaluate the AQM scheme's effectiveness in minimizing packet drops that impact application performance and estimate how well the AQM scheme works with ECN.

2.5. Latency and jitter

The end-to-end latency differs from the queuing delay: it is linked to the network topology and the path characteristics. Moreover, the jitter strongly depends on the traffic and the topology as well. The introduction of an AQM scheme would impact on these metrics and the end-to-end evaluation of performance SHOULD consider them to better assess the AQM schemes.

The guidelines advice that the tester SHOULD determine the minimum, average and maximum measurements for these metrics and the coefficient of variation for their average values as well.

2.6. Discussion on the trade-off between latency and goodput

The metrics presented in this section MAY be considered, in order to discuss and quantify the trade-off between latency and goodput.

This trade-off can also be illustrated with figures following the recommendations of the section 5 of [TCPEVAL2013].

The end-to-end trade-off MUST be considered:

- o end-to-end delay vs. goodput: the x-axis shows the average end-to-end delay and the y-axis the average goodput;
- o drop rate vs. end-to-end delay: the x-axis shows the end-to-end delay and the y-axis the drop rate.

This pair of graphs provide part of a better understanding (1) of the delay/goodput/drop-rate trade-off for a given congestion control mechanism, and (2) of how the goodput and average queue size vary as a function of the traffic load.

3. Generic set up for evaluations

This section presents the topology that can be used for each of the following scenarios, the corresponding notations and discuss various assumptions that have been made in the document.

3.1. Topology and notations

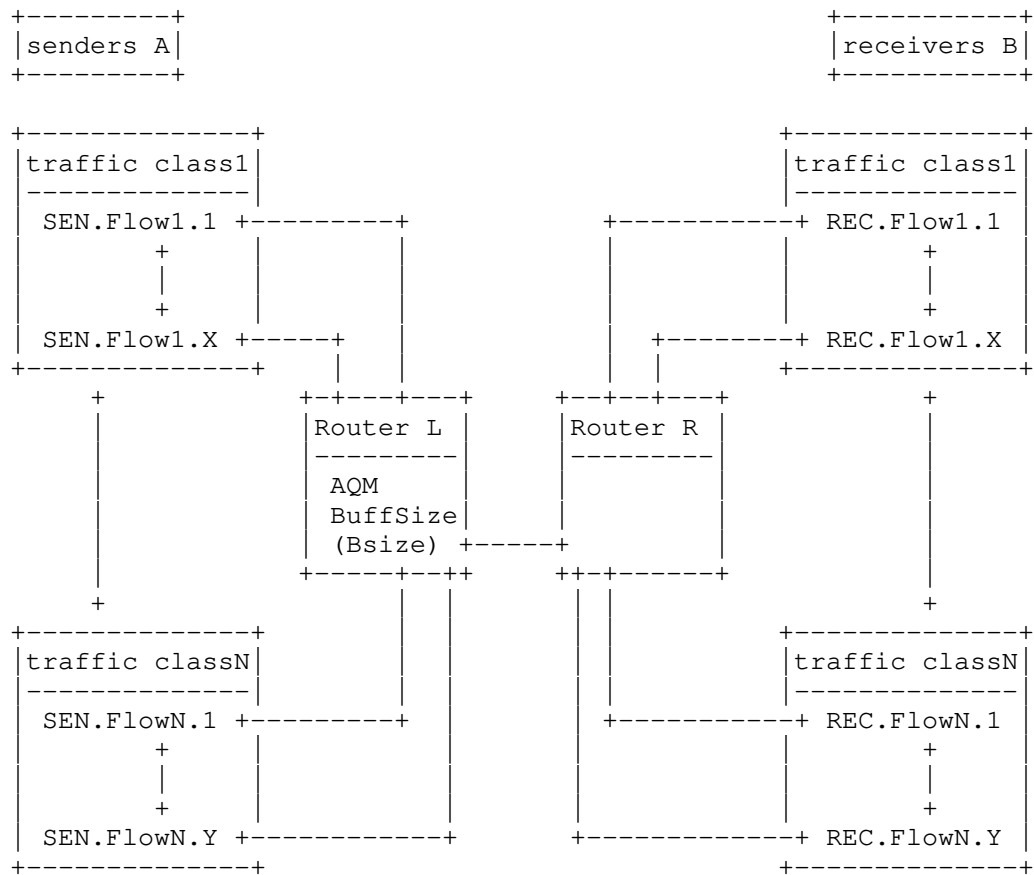


Figure 1 is a generic topology where:

- o various classes of traffic can be introduced;
- o the timing of each flow (i.e., when does each flow start and stop) may be different;
- o each class of traffic can consider various number of flows;
- o each link is characterized by a couple (RTT,capacity);
- o Flows are generated between A and B, sharing a bottleneck (Routers L and R);
- o The links are supposed to be asymmetric in terms of bandwidth: the capacity from senders to receivers is higher than the one from

receivers to senders.

This topology may not perfectly reflect actual topologies, however, this simple topology is commonly used in the world of simulations and small testbeds. This topology can be considered as adequate to evaluate AQM proposals, such as proposed in [TCPEVAL2013]. The tester should pay attention to the topology that has been used to evaluate the AQM scheme against which he compares his proposal.

3.2. Buffer size

The size of the buffers MAY be carefully set considering the bandwidth-delay product. However, if the context or the application requires a specific buffer size, the tester MUST justify and detail the way the maximum queue depth is set while presenting the results of its evaluation. Indeed, the size of the buffer may impact on the AQM performance and is a dimensioning parameter that will be considered for a fair comparison between AQM proposals.

3.3. Congestion controls

This memo features three kind of congestion controls:

- o TCP-friendly congestion controls: a base-line congestion control for this category is TCP New Reno, as explained in [RFC5681].
- o Aggressive congestion controls: a base-line congestion control for this category is TCP Cubic.
- o Less-than Best Effort (LBE) congestion controls: an LBE congestion control 'results in smaller bandwidth and/or delay impact on standard TCP than standard TCP itself, when sharing a bottleneck with it.' [RFC6297]

Recent transport layer protocols are not mentioned in the following sections, for the sake of simplicity.

4. Various TCP variants

Network and end devices need to be configured with a reasonable amount of buffers in order to absorb transient bursts. In some situations, network providers configure devices with large buffers to avoid packet drops and increase goodput. Transmission Control Protocol (TCP) fills up these unmanaged buffers until the TCP sender receives a signal (packet drop) to cut down the sending rate. The larger the buffer, the higher the buffer occupancy, and therefore the queuing delay. On the other hand, an efficient AQM scheme sends out early congestion signals to TCP senders so that the queuing delay is brought under control.

Not all applications run over the same flavor of TCP. Variety of senders generate different classes of traffic which may not react to congestion signals (aka unresponsive flows) or may not cut down their sending rate as expected (aka aggressive flows): AQM schemes aim at maintaining the queuing delay under control, which is challenged if blasting traffics are present.

This section provides guidelines to assess the performance of an AQM proposal based on various metrics presented in Section 2 irrespective of traffic profiles involved -- different senders (TCP variants, unresponsive, aggressive), traffic mix with different applications, etc.

4.1. TCP-friendly Sender

This scenario helps to evaluate how an AQM scheme reacts to a TCP-friendly transport sender. A single long-lived, non application limited, TCP New Reno flow transmits data between sender A and receiver B. Other TCP friendly congestion control schemes such as TCP-friendly rate control [RFC5348] etc MAY also be considered.

For each TCP-friendly transport considered, the graphs described in Section 2.6 MUST be generated.

4.2. Aggressive Transport Sender

This scenario helps to evaluate how an AQM scheme reacts to a transport sender whose sending rate is more aggressive than a single TCP-friendly sender. A single long-lived, non application limited, TCP Cubic flow transmits data between sender A and receiver B. Other aggressive congestion control schemes MAY also be considered.

For each flavor of aggressive transport, the graphs described in Section 2.6 MUST be generated.

4.3. Unresponsive Transport Sender

This scenario helps evaluate how an AQM scheme reacts to a transport sender who is not responsive to congestion signals (ECN marks and/or packet drops) from the AQM scheme. In order to create a test environment that results in queue build up, we consider unresponsive flow(s) whose sending rate is greater than the bottleneck link capacity between routers L and R. Note that faulty transport implementations on end hosts and/or faulty network elements en-route that "hide" congestion signals in packet headers [I-D.ietf-aqm-recommendation] may also lead to a similar situation, such that the AQM scheme needs to adapt to unresponsive traffic.

This scenario consists of long-lived non application limited UDP flow transmits data between sender A and receiver B. Graphs described in Section 2.6 MUST be generated.

4.4. TCP initial congestion window

This scenario helps evaluate how an AQM scheme adapts to a traffic mix consisting of different variants of TCP for various values of the initial congestion window (IW):

- o TCP: Cubic and/or New Reno;
- o IW: 3 or 10 packets.

Figure 2 presents the various cases for the traffic that MUST be generated between sender A and receiver B.

Case	Number of flows			
	Cubic (IW3)	Cubic (IW10)	Reno (IW3)	Reno (IW10)
I	1	1	0	0
II	0	0	1	1
III	1	0	1	0
IV	0	1	0	1

For each of these scenarios, the graphs described in Section 2.6 MUST be generated for each class of traffic.

4.5. Traffic Mix

This scenario helps to evaluate how an AQM scheme reacts to a traffic mix consisting of different applications such as bulk transfer, web, voice, video traffic. These testing cases presented in this subsection have been inspired by the table 2 of [DOCSIS2013]:

- o Bulk TCP transfer
- o Web traffic
- o VoIP
- o Constant bit rate UDP traffic
- o Adaptive video streaming

Figure 3 presents the various cases for the traffic that MUST be generated between sender A and receiver B.

Case	Number of flows				
	VoIP	Webs	CBR	AdaptVid	FTP
I	1	1	0	0	0
II	1	1	0	0	1
III	1	1	0	0	5
IV	1	1	1	0	5
V	1	1	0	1	5

For each of these scenarios, the graphs described in Section 2.6 MUST be generated for each class of traffic. In addition, other metrics such as end-to-end latency, jitter and flow completion time MUST be generated.

5. RTT fairness

5.1. Motivation

The capability of AQM schemes to control the queuing delay highly depends on the way end-to-end protocols react to congestion signals. When the RTT varies, the behaviour of congestion controls is impacted and so the capability of AQM schemes to control the queue. It is therefore important to assess the AQM schemes against a set of RTTs (e.g., from 5 ms to 200 ms).

Also, asymmetry in terms of RTT between various paths SHOULD be considered so that the fairness between the flows can be discussed as one may react faster to congestion than another. The introduction of AQM schemes may improve this fairness.

Moreover, introducing an AQM scheme may result in the absence of fairness between the flows, even when the RTTs are identical. This potential lack of fairness SHOULD be evaluated.

5.2. Required tests

The topology that SHOULD be used is detailed in Figure 1:

- o to evaluate the inter-RTT fairness, for each run, ten flows divided into two categories. Category I (Flow1.1, ..., Flow1.5) which RTT between sender A and Router L SHOULD be 5ms. Category II (Flow2.1, ..., Flow 2.5) which RTT between sender A and Router

L SHOULD be in [5ms;200ms].

- o to evaluate the impact of the RTT value on the AQM performance and the intra-protocol fairness, for each run, ten flows (Flow1.1, ..., Flow1.5 and Flow2.1, ..., Flow2.5) SHOULD be introduced. For each experiment, the set of RTT SHOULD be the same for all the flows and in [5ms;200ms].

These flows MUST use the same congestion control algorithm.

5.3. Metrics to evaluate the RTT fairness

The output that MUST be measured is:

- o for the inter-RTT fairness: (1) the cumulated average goodput of the flows from Category I, `goodput_Cat_I` (Section 2.4); (2) the cumulated average goodput of the flows from Category II, `goodput_Cat_II` (Section 2.4); (3) the ratio `goodput_Cat_II/goodput_Cat_I`; (4) the average packet drop rate for each category (Section 2.2).
- o for the intra-protocol RTT fairness: (1) the cumulated average goodput of the ten flows (Section 2.4); (2) the average packet drop rate for the ten flows (Section 2.2).

6. Burst absorption

6.1. Motivation

Packet arrivals can be bursty due to various reasons. Dropping one or more packets from a burst may result in performance penalties for the corresponding flows since the dropped packets have to be retransmitted. Performance penalties may turn into unmet SLAs and be disincentives to AQM adoption. Therefore, an AQM scheme SHOULD be designed to accommodate transient bursts. AQM schemes do not present the same tolerance to bursts of packets arriving in the buffer: this tolerance MUST be quantified.

Note that accommodating bursts translates to higher queue length and queuing delay. Naturally, it is important that the AQM scheme brings bursty traffic under control quickly. On the other hand, spiking packet drops in order to bring packet bursts quickly under control could result in multiple drops per flow and severely impact transport and application performance. Therefore, an AQM scheme SHOULD bring bursts under control by balancing both aspects -- (1) queuing delay spikes are minimized and (2) performance penalties for ongoing flows in terms of packet drops are minimized.

An AQM scheme maintains short queues to allow the remaining space in the queue for bursts of packets. The tolerance to bursts of packets depends on the number of packets in the queue, which is directly linked to the AQM algorithm. Moreover, one AQM scheme may implement a feature controlling the maximum size of accepted bursts, that may depend on the buffer occupancy or the currently estimated queuing delay. Also, the impact of the buffer size on the burst allowance MAY be evaluated.

6.2. Required tests

For this scenario, the following traffic MUST be generated from sender A to receiver B:

- o IW10: TCP transfer with initial congestion window set to 10 of 5MB;
- o Bursty video frames;
- o Web traffic;
- o Constant bit rate UDP traffic.

Figure 4 presents the various cases for the traffic that MUST be generated between sender A and receiver B.

Case	Number of traffic			
	Video	Webs	CBR	Bulk Traffic (IW10)
I	0	1	1	0
II	0	1	1	1
III	1	1	1	0
IV	1	1	1	0
V	1	1	1	1

6.3. Metrics to evaluate the burst absorption capacity

For each of these scenarios, the graphs described in Section 2.6 MUST be generated. In addition, other metrics such as end-to-end latency, jitter, flow completion time MUST be generated.

7. Stability

7.1. Motivation

Network devices experience varying operating conditions depending on factors such as time of day, deployment scenario etc. For example:

- o Traffic and congestion levels are higher during peak hours than off-peak hours.
- o In the presence of scheduler, a queue's draining rate may vary depending on other queues: a low load on a high priority queue implies higher draining rate for lower priority queues.
- o The available capacity on the physical layer may vary over time such as in the context of lossy channels.

Whether the target context is a not stable environment, the capability of an AQM scheme to actually maintain its control on the queuing delay and buffer occupancy is challenged. This document propose guidelines to assess the behaviour of AQM schemes under varying congestion levels and varying draining rates.

7.2. Required tests

7.2.1. Mild Congestion

This scenario helps to evaluate how an AQM scheme reacts to a light load of incoming traffic resulting in mild congestion -- packet drop rates less than 1%. Each single-lived non application limited TCP flow transfers data.

For this scenario, the graphs described in Section 2.6 MUST be generated.

7.2.2. Medium Congestion

This scenario helps to evaluate how an AQM scheme reacts to incoming traffic resulting in medium congestion -- packet drop rates between 1%-3%. Each single-lived non application limited TCP flow transfers data.

For this scenario, the graphs described in Section 2.6 MUST be generated.

7.2.3. Heavy Congestion

This scenario helps to evaluate how an AQM scheme reacts to incoming traffic resulting in heavy congestion -- packet drop rates between 5%-10%. Each single lived non application limited TCP flow transfers data.

For this scenario, the graphs described in Section 2.6 MUST be generated.

7.2.4. Varying Available Bandwidth

This scenario helps evaluate how an AQM scheme adapts to varying available bandwidth on the outgoing link.

To simulate varying draining rates, the bottleneck bandwidth between nodes 'Router L' and 'Router R' varies over the course of the experiment as follows:

- o Experiment 1: the capacity varies between two values according to a large time scale. As an example, the following phases may be considered: phase I - 100Mbps during 0-5s; phase II - 10Mbps during 5-10s; phase I again, ... and so on.
- o Experiment 2: the capacity varies between two values according to a short time scale. As an example, the following phases may be considered: phase I - 100Mbps during 100ms; phase II - 10Mbps during 100ms; phase I again during 100ms, ... and so on.

More realistic fluctuating bandwidth patterns MAY be considered.

The scenario consists of TCP New Reno flows between sender A and receiver B. In order to better assess the impact of draining rates on the AQM behavior, the tester MUST compare its performance with those of drop-tail.

For this scenario, the graphs described in Section 2.6 MUST be generated. Moreover, one graph SHOULD be generated for each of the phases previously detailed.

7.3. Parameter sensitivity and stability analysis

An AQM scheme's control law is the primary means by which the AQM controls queuing delay. Hence understanding the AQM control law is critical to understanding AQM behavior. The AQM's control law may include several input parameters whose values affect the AQM output behavior and stability. Additionally, AQM schemes may auto-tune parameter values in order to maintain stability under different network conditions (such as different congestion levels, draining rates or network environments). The stability of these auto-tuning techniques is also important to understand.

AQM proposals SHOULD provide background material showing control theoretic analysis of the AQM control law and the input parameter space within which the control law operates as expected; or could use other ways to discuss its stability. For parameters that are auto-tuned, the material SHOULD include stability analysis of the auto-tuning mechanism(s) as well. Such analysis helps to understand an AQM's control law better and the network conditions/deployments under which the AQM is stable.

8. Implementation cost

8.1. Motivation

An AQM's successful deployment is directly related to its ease of implementation. Network devices may need hardware or software implementations of the AQM. Depending on a device's capabilities and limitations, the device may or may not be able to implement some or all parts of the AQM logic.

AQM proposals SHOULD provide pseudo-code for the complete AQM scheme, highlighting generic implementation-specific aspects of the scheme such as "drop-tail" vs. "drop-head", inputs (current queuing delay, queue length), computations involved, need for timers etc. This helps identify costs associated with implementing the AQM on a particular hardware or software device. Also, it helps the WG understand which kind of devices can easily support the AQM and which cannot.

8.2. Required discussion

AQM proposals SHOULD highlight parts of AQM logic that are device dependent and discuss if and how AQM behavior could be impacted by the device. For example, a queue-delay based AQM scheme requires current queuing delay as input from the device. If the device already maintains this value, then it is trivial to implement the AQM logic on the device. On the other hand, if the device provides indirect means to estimate queuing delay (for example: timestamps, dequeuing rate etc.), then the AQM behavior is sensitive to how good the queuing delay estimate turns out on that device. Highlighting the AQM's sensitivity to queuing delay estimate helps implementers identify optimal means of implementing the AQM on a device.

9. Operator control knobs and auto-tuning

One of the biggest hurdles for RED deployment was/is its parameter sensitivity to operating conditions -- how difficult it is to tune important RED parameters for a deployment in order to get maximum benefit from the RED implementation. Fluctuating congestion levels and network conditions add to the complexity. Incorrect parameter values lead to poor performance. This is one reason why RED is reported to be usually turned off.

Any AQM scheme is likely to have parameters whose values affect the AQM's control law and behavior. Exposing all these parameters as control knobs to a network operator (or user) can easily result in an unsafe AQM deployment. Unexpected AQM behavior ensues when parameter values are not set properly. A minimal number of control knobs minimizes the number of ways a, possible naive, user can break the AQM system. Fewer control knobs make the AQM scheme more user-friendly and easier to deploy and debug.

We recommend that an AQM scheme SHOULD minimize the number of control knobs exposed for operator tuning. An AQM scheme SHOULD expose only those knobs that control the macroscopic AQM behavior such as queue delay threshold, queue length threshold, etc.

Additionally, an AQM scheme's safety is directly related to its stability under varying operating conditions such as varying traffic profiles and fluctuating network conditions, as described in Section 7. Operating conditions vary often and hence it is necessary that the AQM MUST remain stable under these conditions without the need for additional external tuning. If AQM parameters require tuning under these conditions, then the AQM MUST self-adapt necessary parameter values by employing auto-tuning techniques.

10. Interaction with ECN

10.1. Motivation

Apart from packet drops, Explicit Congestion Notification (ECN) is an alternative means to signal data senders about network congestion. The AQM recommendation document [I-D.ietf-aqm-recommendation] describes some of the benefits of using ECN with AQM.

10.2. Required discussion

An AQM scheme MAY support ECN, in which case testers MUST discuss and describe the support of ECN.

11. Interaction with scheduling

11.1. Motivation

Coupled with an AQM scheme, a router may schedule the transmission of packets in a specific manner by introducing a scheduling scheme. This algorithm may create sub-queues and integrate a dropping policy on each of these sub-queues. Another scheduling policy may modify the way packets are sequenced, modifying the timestamp of each packet.

11.2. Required discussion

The scheduling and the AQM conjointly impact on the end-to-end performance. During the characterization process of a dropping policy, the tester MAY discuss the feasibility to add scheduling on top of its algorithm. This discussion MAY detail if the dropping policy is applied while packets are enqueued or dequeued.

12. Discussion on methodology, metrics, AQM comparisons and packet sizes

12.1. Methodology

A sufficiently detailed description of the test setup SHOULD be provided. Indeed, that would allow other to replicate the tests if needed. This test setup MAY include software and hardware versions. The tester MAY make its data available.

The proposals SHOULD be experimented on real systems, or they MAY be evaluated with event-driven simulations (such as NS-2, NS-3, OMNET, etc.). The proposed scenarios are not bound to a particular evaluation toolset.

12.2. Comments on metrics measurement

In this document, we present the end-to-end metrics that SHOULD be evaluated to evaluate the trade-off between latency and goodput. The queue-related metrics enable a better understanding of the AQM behavior under tests and the impact of its internal parameters. Whenever it is possible, these guidelines advice to consider queue-related metrics, such as link utilization, queuing delay, queue size or packet loss.

These guidelines could hardly detail the way the metrics can be measured depends highly on the evaluation toolset.

12.3. Comparing AQM schemes

This memo recognizes that the guidelines mentioned above may be used for comparing AQM schemes. This memo recommends that AQM schemes MUST be compared against both performance and deployment categories. In addition, this section details how best to achieve a fair comparison of AQM schemes by avoiding certain pitfalls.

12.3.1. Performance comparison

AQM schemes MUST be compared against all the generic scenarios presented in this memo. AQM schemes MAY be compared for specific network environments such as data center, home networks etc. If an AQM scheme's parameter(s) were externally tuned for optimization or other purposes, these values MUST be disclosed.

Note that AQM schemes belong to different varieties such as queue-length based scheme (ex: RED) or queue-delay based scheme (ex: CoDel, PIE). Also, AQM schemes expose different control knobs associated with different semantics. For example, while both PIE and CoDel are queue-delay based schemes and each expose a knob to control the queueing delay -- PIE's "queueing delay reference" vs. CoDel's "queueing delay target", the two schemes' knobs have different semantics resulting in different control points. Such differences in AQM schemes can be easily overlooked while making comparisons.

This document recommends the following procedures for a fair performance comparison of two AQM schemes:

1. comparable control parameters and comparable input values: carefully identify the set of parameters that control similar behavior between the two AQM schemes and ensure these parameters have comparable input values. For example, while comparing how well a queue-length based AQM X controls queueing delay vs. queue-delay based AQM Y, identify the two schemes' parameters that control queue delay and ensure that their input values are comparable. Similarly, to compare two AQM schemes on how well they accommodate bursts, identify burst-related control parameters and ensure they are configured with similar values.
2. compare over a range of input configurations: there could be situations when the set of control parameters that affect a specific behavior have different semantics between the two AQM schemes. As mentioned above, PIE's knob to control queue delay has different semantics from CoDel's. In such situations, the schemes MUST be compared over a range of input configurations. For example, compare PIE vs. CoDel over the range of delay input configurations -- 5ms, 10ms, 15ms etc.

12.3.2. Deployment comparison

AQM schemes MUST be compared against deployment criteria such as the parameter sensitivity (Section 7.3), the auto-tuning (Section 9) or the implementation cost (Section 8).

12.4. Packet sizes and congestion notification

An AQM scheme may be considering packet sizes while generating congestion signals. [RFC7141] discusses the motivations behind the same. For example, control packets such as DNS requests/responses, TCP SYN/ACKs are small, and their loss can severely impact application performance. An AQM scheme may therefore be biased towards small packets by dropping them with smaller probability compared to larger packets. However, such an AQM scheme is unfair to data senders generating larger packets. Data senders, malicious or otherwise, are motivated to take advantage of the AQM scheme by transmitting smaller packets, and could result in unsafe deployments and unhealthy transport and/or application designs.

An AQM scheme SHOULD adhere to recommendations outlined in [RFC7141], and SHOULD NOT provide undue advantage to flows with smaller packets.

13. Acknowledgements

This work has been partially supported by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700).

14. Contributors

Many thanks to S. Akhtar, A.B. Bagayoko, F. Baker, D. Collier-Brown, G. Fairhurst, T. Hoiland-Jorgensen, C. Kulatunga, R. Pan, D. Taht and M. Welzl for detailed and wise feedback on this document.

15. IANA Considerations

This memo includes no request to IANA.

16. Security Considerations

This document, by itself, presents no new privacy nor security issues.

17. References

17.1. Normative References

[I-D.ietf-aqm-recommendation]
Baker, F. and G. Fairhurst, "IETF Recommendations Regarding Active Queue Management", Internet-Draft draft-ietf-aqm-recommendation-07, August 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, 1997.

[RFC7141] Briscoe, B. and J. Manner, "Byte and Packet Congestion Notification", RFC 7141, 2014.

17.2. Informative References

[BB2011] "BufferBloat: what's wrong with the internet?", ACM Queue vol. 9, 2011.

[CODEL] Nichols, K. and V. Jacobson, "Controlling Queue Delay", ACM Queue , 2012.

[DOCSIS2013]
White, G. and D. Rice, "Active Queue Management Algorithms for DOCSIS 3.0", Technical report - Cable Television Laboratories , 2013.

[LOSS-SYNCH-MET-08]
Hassayoun, S. and D. Ros, "Loss Synchronization and Router Buffer Sizing with High-Speed Versions of TCP", IEEE INFOCOM Workshops , 2008.

[PIE] Pan, R., Natarajan, P., Piglione, C., Prabhu, MS., Subramanian, V., Baker, F. and B. VerSteeg, "PIE: A lightweight control scheme to address the bufferbloat problem", IEEE HPSR , 2013.

- [RFC2309] Braden, B., Clark, D.D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K.K., Shenker, S., Wroclawski, J. and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, April 1998.
- [RFC5348] Floyd, S., Handley, M., Padhye, J. and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 5348, September 2008.
- [RFC5681] Allman, M., Paxson, V. and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.
- [RFC6297] Welzl, M. and D. Ros, "A Survey of Lower-than-Best-Effort Transport Protocols", RFC 6297, June 2011.
- [TCPEVAL2013] Hayes, D., Ros, D., Andrew, L.L.H. and S. Floyd, "Common TCP Evaluation Suite", IRTF ICCRG, 2013.
- [YU06] Jay, P., Fu, Q. and G. Armitage, "A preliminary analysis of loss synchronisation between concurrent TCP flows", Australian Telecommunication Networks and Application Conference (ATNAC) , 2006.

Authors' Addresses

Nicolas Kuhn, editor
Telecom Bretagne
2 rue de la Chataigneraie
Cesson-Sevigne, 35510
France

Phone: +33 2 99 12 70 46
Email: nicolas.kuhn@telecom-bretagne.eu

Preethi Natarajan, editor
Cisco Systems
510 McCarthy Blvd
Milpitas, California
United States

Email: prenatar@cisco.com

David Ros
Simula Research Laboratory AS
P.O. Box 134
Lysaker, 1325,
Norway

Phone: +33 299 25 21 21
Email: dros@simula.no

Naeem Khademi
University of Oslo
Department of Informatics, PO Box 1080 Blindern
N-0316 Oslo,
Norway

Phone: +47 2285 24 93
Email: naeemk@ifi.uio.no

Active Queue Management
and Packet Scheduling (aqm)
Internet Draft
Intended status:

Expires: November 2016

Wolfram Lautenschlaeger
Nokia
Bell Labs

May 25, 2016

Global Synchronization Protection for Packet Queues
draft-lauten-aqm-gsp-03.txt

Abstract

The congestion avoidance processes of transmission capacity sharing TCP flows tend to be synchronized among each other, so that the rate variations of the individual flows do not compensate. In contrary, they accumulate into large variations of the whole aggregate. The effect is known as global synchronization. Large queuing buffer demand and large latency and jitter are the consequences. Global Synchronization Protection (GSP) is an extension of regular tail drop packet queuing schemes that prevents global synchronization. For large traffic aggregates the de-correlation between the individual flow variations reduces buffer demand and packet sojourn time by an order of magnitude and more. Even though quite simple, the solution has a theoretical background and is not heuristic. It has been tested with a Linux kernel implementation and shows equivalent performance as other relevant AQM schemes.

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>

This Internet-Draft will expire on November 25, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

1. Introduction.....	2
2. Conventions used in this document.....	4
3. Root cause of global synchronization.....	4
4. Protecting queues of global synchronization.....	5
4.1. Basic algorithm.....	5
4.2. Interval adaptation at large flow numbers.....	5
4.3. Interval adaptation at small RTT.....	7
4.4. Sanity checks and special cases.....	7
5. Delay based operation.....	8
5.1. Queue delay vs. queue size.....	8
5.2. Delay based GSP.....	8
6. Security Considerations.....	9
7. IANA Considerations.....	9
8. Conclusions.....	9
9. References.....	9
9.1. Normative References.....	9
9.2. Informative References.....	9
10. Acknowledgments.....	10

1. Introduction

The congestion window (CWND) of a particular TCP connection, in combination with the round trip time (RTT), limits the transmission rate of the flow, which enables adaptation of the sending rate to the actual network conditions, [1]. TCP uses a rather coarse congestion control feedback by halving the congestion window in response to

packet loss. To fill a bottleneck link by 100% anyway, a packet buffer in front of the link is required. For a single TCP flow a buffer in the range of bottleneck capacity multiplied by the round trip time is required (bandwidth-delay product rule, BDP), [2]. For aggregated traffic of many flows the picture is not so clear. Conservative estimations tend towards BDP of the whole aggregate, i.e. link capacity * RTT. At the other hand, rate reductions due to CWND halving are still only in the range of a particular flow rate. With the assumption of N sharing flows, this yields ideally a buffer size of only (link capacity/N)*RTT. Unfortunately this value cannot be reached in practice. It would require a uniform distribution of rate reductions by the different flows over time. In opposite, rate reductions of bottleneck sharing flows tend to synchronize among each other, which is called global synchronization. In worst case, with all flows synchronized, the buffer demand is back at BDP of the whole traffic, thus confirming the conservative estimation.

There are cases where global synchronization does not occur, in particular large number of flows ($N > 500$), large spread of RTT between the different flows, and high frequency of flow renewals. In these cases the buffer size can be reduced to BDP/\sqrt{N} , which lies between the conservative and overly optimistic estimations above, [3]. Nevertheless there are still doubts, whether the absence of global synchronization is a general reliable design assumption for high capacity links, [4].

Most Active Queue Management (AQM) algorithms are aiming at better control of the queue size (RED [5]) or the queue delay (CoDel [6], PIE [7]), which implies control over global synchronization. Global Synchronization Protection (GSP [10]) goes the other way round. It suppresses the root cause of global synchronization and de-correlates the CWND variations of the competing flows, but it does not try to impact the behavior of a particular flow. This way it moves the buffer size demand down from conservative BDP of the whole link into the direction towards the ideal BDP of only one of the competing flows.

Experiments on GSP performance, as reported in [10], show that the stabilizing effect of GSP is equivalent to that of the other AQMs. It is a simple extension to plain tail drop queues. The basic algorithm is memoryless and does not need artificial randomization. Particularly for small numbers of flows it performs better than randomized AQMs like e.g. RED or PIE. Without any special precautions GSP is robust to bursts and never drops at low or empty queues. Automatic parameter adaptation, if needed at all, is external to the basic control loop, which makes it less time critical and less

sensitive to traffic conditions than other AQMs with integrated control and adaptation loops.

2. Conventions used in this document

In this document, the term "packet drop" is used for congestion notification, silently assuming that congestion marking for ECN could be equally applied.

In this document, the term "queue size" is preferably applied in number of bytes, however, the algorithm could be also applied to the number of packets, or even to the queuing delay (milliseconds).

3. Root cause of global synchronization

Global synchronization occurs in cases where a number of greedy TCP flows with comparably uniform RTT cross a tail drop queue in front of a shared transmission link. Greedy TCP means, the flow is probing the available capacity on this particular link and is not limited elsewhere (up- or downstream). Tail drop means, a newly arriving packet is placed at the end of the queue if buffer space permits. Otherwise it is dropped. The queue is drained from head of the queue at the speed of the link as long as packets are available.

In congestion avoidance state, all senders gradually increase their sending rate, which is, after a while, exceeding the link capacity so that the queue in front of the link is filling up. At some point in time, a first packet is dropped due to lack of buffer space. Ideally, the TCP flow, where the dropped packet belongs to, reduces its sending rate, the queue relaxes, its size goes down, and subsequently arriving packets again can be placed in the buffer. Senders continue to increase their sending rates until the next drop, and so on.

Unfortunately not one, but several packets get dropped in such incident for following reason: The rate reduction due to the first dropped packet needs at least one RTT to take effect at the queue entry. During that RTT interval all senders continue to gradually increase their sending rates, whereas the queue is still full. Further packets need to be dropped. It can be shown analytically that for N flows with NewReno and delayed ACK the number of drops is in the range of $N/2$. Experiments confirm this and show an even higher number with CUBIC. The outcome is that even though the rate reduction by one flow would suffice, not one, but as much as half of the flows are triggered within one RTT to reduce their sending rates - we have global synchronization. A more detailed analytical and experimental investigation of the effect can be found in [8].

4. Protecting queues of global synchronization

4.1. Basic algorithm

The basic algorithm works as follows: Set a threshold on queue size below the actual buffer size. If a new packet arrives and the queue size is above the threshold, then immediately drop that packet. After that, ignore any further threshold violation for a timeout interval of 1 - 3 RTT. After expiry of the timeout proceed as above.

Algorithm:

```
initialization:
    interval = e.g. 2 * RTT
    threshold = e.g. 1/2 * buffer size
    timeout_expiry = now(), with now() returning the current time

at any packet arrival do:
    if queue_size > threshold && now() > timeout_expiry:
        drop this packet
        timeout_expiry = now() + interval
    else
        enqueue this packet
```

The first dropped packet is triggering the rate reduction by one of the end points. During the timeout the queue is growing further beyond the threshold until the rate reduction takes effect at queue entry. Afterwards the queue size should have dropped below the threshold, so that at timeout expiry the threshold is typically not violated anymore. No explicit action occurs at timeout expiry, which makes the parameter rather insensitive to the actual traffic characteristics. Even if the timeout interval is too short, the algorithm still reduces global synchronization.

4.2. Interval adaptation at large flow numbers

The basic algorithm works well for moderate numbers of flows N , i.e. in a range of $1 < N < 20$. More precisely, at flow numbers N smaller than the average CWND of one of the sharing flows. At larger numbers the total rate increase during the timeout interval is larger than the subsequent rate reduction by one of the flows. As consequence, after timeout expiry the threshold is still violated, the queue is growing further and further, and, eventually, reaches the buffer limit and enters tail drop operation. The performance is still better

than plain tail drop and one could rely on the observation that at large flow numbers global synchronization disappears, anyway.

Alternatively the initial timeout interval can be reduced, depending on the actual traffic, in a way, where not just once, but twice, or even more times per RTT the timeout expires. The adaptation criterion is the proportion of time above and below threshold. In regular operation according to the basic algorithm, the queue is most of the time below the threshold. If, however, the queue is more frequently above than below threshold, the interval should be reduced until equilibrium is reached. In this condition the queue is oscillating around the threshold, periodically dropping during times above the threshold, quite similar like PED [9].

Algorithm:

initialization:

```
    tau = a preset parameter controlling the adaptation speed;  
          e.g. 500 milliseconds or 5 * initial_interval  
    initial_interval - the preset timeout interval as in the basic  
                      algorithm  
    cumTime = 0; the cumulative time above/below threshold  
    state = CLEAR; the recent overflow state of the queue
```

at any packet arrival do:

```
    if the packet would overflow the buffer (hard tail drop):  
        state = OVERFLOW  
    if state == OVERFLOW && queue is empty:  
        state = DRAIN  
    if state == DRAIN && queue_size > threshold:  
        state = CLEAR
```

update the cumulative time cumTime:

```
    account by twice the duration for queue episodes  
        that are entirely ABOVE the threshold  
    if status == CLEAR:  
        account by negated duration for queue episodes  
            that are entirely BELOW the threshold
```

clamp the cumulative time:

```
    cumTime = max(cumTime, 0)  
    cumTime = min(cumTime, some sanity limit  
                  of several minutes)
```

```
calculate timeout interval (to be used at next drop decision):  
    interval = initial_interval/(1+cumTime/tau)
```

The adaptation heuristics works as follows: The basic GSP algorithm executes single drops per threshold violation as long as the cumulative time (cumTime) is clamped at zero. As soon as the cumulative time gets positive, the adaptation algorithm implements an integral controller for the drop rate that the basic GSP algorithm executes during times of threshold violation. The transition between both operating conditions is smooth. The adaptation speed is controlled by the parameter tau.

Plain adaptation by cumulative times above / below threshold might latch up in circumstances where abrupt traffic changes cause massive buffer overflows. To avoid this, after a hard buffer overflow the accounting for times below threshold is suspended until the queue performed a full cycle of down to empty and back above threshold.

4.3. Interval adaptation at small RTT

The RTT is not known exactly but there should be at least a rough idea on the range of RTT for setting up the timeout interval. If this estimation is much too large, a similar situation occurs like in the large flow numbers case. The total rate increase during the timeout interval (which turns out to be multiple RTTs) is larger than the subsequent rate reduction by one flow. The adaptation rule is the same as for large flow numbers, section 4.2.

4.4. Sanity checks and special cases

An additional rule can be introduced that prevents large packet bursts from immediately triggering the drop: Restart the timeout not only after a packet drop but also whenever a packet is arriving at an empty queue.

```
at any packet arrival do:  
    if queue is empty:  
        timeout_expiry = now() + interval
```

5. Delay based operation

5.1. Queue delay vs. queue size

Recent new AQM proposals ([6], [7]) are focusing on queue delay rather than on queue size in bytes. One reason for this move is that ideally the steady state queue oscillation depends only on the RTT and on the number of sharing TCP flows – if measured in delay. If measured in bytes, however, the queue oscillation additionally depends on the link capacity. The oscillation span sets the minimum queue size for 100% link utilization. (This is where the bandwidth delay product rule comes from.) A larger queue creates only unnecessary delay (standing queue).

Obviously it is preferable to stabilize the delay instead of the size. It eliminates the interface rate from the parameter list, which is particularly welcome in circumstances with unknown or variable drain rate. Such situations are typical for low priority queues in front of a priority scheduler and generally in wireless scenarios.

5.2. Delay based GSP

In section 4. we silently assumed queue size in bytes. However, the algorithm can be equally applied to the queue delay (packet sojourn time). In this case the threshold has to be in milliseconds, whereas the empty queue condition remains the same as before.

While the queue size in bytes or packets is typically maintained by ordinary queue implementations, obtaining the queue delay requires additional effort. Two solutions are available and both are applicable to GSP: Time stamping of packets like in CoDel [6] or estimating the drain rate for a translation of size into delay like in PIE [7].

Algorithm (time stamping):

```
at any arrival of a packet p do:
    p.time = now()

at any departure of a packet p do:
    queue_delay = now() - p.time
```

The basic algorithm of section 4.1. rephrased to delay based operation:

```
at any packet arrival do:
    if queue_delay > threshold && now() > timeout_expiry:
        drop this packet
        timeout_expiry = now() + interval
    else
        enqueue this packet
```

Please note that `queue_delay` is a per queue variable, not per packet, i.e. the drop decision at enqueueing (tail drop) depends on the delay that another, most recently dequeued packet experienced. This approach is a forecast of the expected delay and it is justified by the inherent inrtance of the queue itself.

6. Security Considerations

Global synchronization is a particular problem of many elastic flows sharing a bottleneck. GSP is there to prevent this. But it does not protect of unresponsive flows. If the congestion notification according to section 4.1. randomly hits an unresponsive flow then the expected rate reduction within the timeout interval might simply not happen, which postpones the notification by one timeout interval. The interval adaptation of section 4.2. automatically ensures that the timeout interval is reduced accordingly, depending on the average fraction of unresponsive traffic. In extreme cases, when the unresponsive traffic alone is exceeding the link capacity, GSP behaves like plain tail drop.

7. IANA Considerations

There are no actions for IANA.

8. Conclusions

tbc

9. References

9.1. Normative References

9.2. Informative References

- [1] Van Jacobson, Congestion avoidance and control, Proc. SIGCOMM '88, 1988

- [2] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm, *Comput. Commun. Rev.*, 27.3, 1997, pp. 67-82.
- [3] G. Appenzeller, I. Keslassy, and N. McKeown, Sizing router buffers, *Proc. ACM SIGCOMM '04*, 2004.
- [4] G. Vu-Brugier, R. S. Stanojevic, D. J. Leith, R. N. Shorten, A critique of recently proposed buffer-sizing strategies, *ACM SIGCOMM Computer Communication Review*, 37.1, 2007
- [5] S. Floyd, Van Jacobsen, Random Early Detection Gateways for Congestion Avoidance, *IEEE/ACM Trans. Networking*, 1.4, 1993
- [6] K. Nichols, Van Jacobson, "Controlling Queue Delay", *ACM Queue - Networks*, 2012
- [7] R. Pan, P. Natarajan, C. Piglione, M.S. Prabhu, V. Subramanian, F. Baker, B. VerSteeg, PIE: A lightweight control scheme to address the bufferbloat problem, 14th High Performance Switching and Routing (HPSR), 2013 IEEE
- [8] W. Lautenschlaeger, A deterministic TCP bandwidth sharing model, 2014, online <http://arxiv.org/pdf/1404.4173v1>
- [9] A. Francini, Beyond RED: Periodic Early Detection for On-Chip Buffer Memories in Network Elements, *Proc. IEEE High-Performance Switching and Routing Conference (HPSR 2011)*, Cartagena, Spain, July 4-6, 2011
- [10] W. Lautenschlaeger, A. Francini, Global Synchronization Protection for Bandwidth Sharing TCP Flows in High-Speed Links, *IEEE HPSR 2015*, Budapest, Hungary, July 2015, online <http://arxiv.org/pdf/1602.05333>

10. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Wolfram Lautenschlaeger
Alcatel-Lucent Deutschland AG
Bell Labs
Lorenzstrasse 10
70435 Stuttgart
Germany

Email: Wolfram.Lautenschlaeger@nokia.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 10, 2015

M. Welzl
University of Oslo
G. Fairhurst
University of Aberdeen
March 09, 2015

The Benefits to Applications of using Explicit Congestion Notification
(ECN)
draft-welzl-ecn-benefits-02

Abstract

This document describes the potential benefits to applications when they enable Explicit Congestion Notification (ECN). It outlines the principal gains in terms of increased throughput, reduced delay and other benefits when ECN is used over network paths that include equipment that supports ECN-marking. The focus of this document is on usage of ECN, not its implementation in hosts, routers and other network devices.

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 <http://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 September 10, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

1. Introduction

Internet Transports (such as TCP and SCTP) have two ways to detect congestion: the loss of a packet and, if Explicit Congestion Notification (ECN) [RFC3168] is enabled, by reception of a packet with a Congestion Experienced (CE)-marking in the IP header. Both of these are treated by transports as indications of (potential) congestion. ECN may also be enabled by other transports: UDP applications may enable ECN when they are able to correctly process the ECN signals [RFC5405] (e.g. ECN with RTP [RFC6679]).

A network device (router, middlebox, or other device that forward packets through the network) that does not support AQM, typically uses a drop-tail policy to discard excess IP packets when its queue becomes full. The discard of this packet serves as a signal to the end-to-end transport that there may be congestion on the network path being used. This triggers a congestion control reaction to reduce the maximum rate permitted by the sending endpoint.

When an application uses a transport that enables the use of ECN, the transport layer sets the ECT(0) or ECT(1) codepoint in the IP header of packets that it sends. This indicate to network devices that they may mark, rather than drop, packets in periods of congestion. This marking is generally performed by Active Queue Management (AQM) [RFC2309.bis] and may be the result of various AQM algorithms, where the exact combination of AQM/ECN algorithms is generally not known by the transport endpoints. The focus of this document is on usage of ECN, not its implementation in hosts, routers and other network devices.

ECN makes it possible for the network to signal congestion without packet loss. This lets the network deliver some packets to an application that would otherwise have been dropped. This reduction in packet loss is the most obvious benefit of ECN, but it is often relatively modest. However, enabling ECN can also result in a number of beneficial side-effects, some of which may be much more significant than the immediate reduction in packet loss from ECN-marking instead of dropping packets.

The remainder of this document discusses the potential for ECN to positively benefit an application without making specific assumptions about configuration or implementation.

[RFC3168] describes a method in which a router, sets the CE codepoint of an ECN-Capable packet at the time that the network device would otherwise have dropped the packet. While it has often been assumed that network devices mark packets at the same level of congestion at which they would otherwise have dropped them (e.g., when a queue reaches an AQM threshold), separate configuration of the drop and mark thresholds is known to be supported in some network devices and this is recommended in [RFC2309.bis]. Some benefits of ECN that are discussed rely upon routers marking packets at a lower level of congestion before they would otherwise drop packets [KH13].

Some benefits are also only realised when the transport endpoint behaviour is also updated, this is discussed further in Section 5.

2. ECN Deployment

For an application to use ECN requires that the endpoint first enables ECN within the transport. This requires network devices along the path to at least pass IP packets that set ECN codepoints, and do not drop packets because these codepoints are used. This is the recommended behaviour for network devices [RFC2309.bis]. Applications and transports (such as TCP or SCTP) can be designed to fall-back to not using ECN when they discover they are using a path that does not allow use of ECN (e.g., a firewall or other network device configured to drop the ECN codepoint).

XXX NOTE: A future revision could include some words and reference a paper on the current state of network support for transparently passing the ECN codepoints.

For an application at an endpoint to gain benefit from ECN, network devices need to enable ECN marking. Not all network devices along the path need to enable ECN, for the application to benefit. Any network devices that does not set a CE-codepoint can be expected to drop packets under congestion. Applications that experience congestion in such endpoints do not see any benefit from using ECN, but would see benefit if the congestion were to occur within a network device that did support ECN.

ECN can be incrementally deployed in the general Internet. The IETF has provided guidance on configuration and usage in [RFC2309.bis].

ECN may also be deployed within a controlled environment, for example within a data centre or within a well-managed private network. In this case, the use of ECN may be tuned to the specific use-case. An example is Datacenter TCP (DCTCP) [AL10].

Deployment needs also to consider the requirements for processing ECN at tunnel endpoints of network tunnels, and guidance on the treatment of ECN is provided in [RFC6040]. Further guidance on the encapsulation and use of ECN by non-IP network devices is provided in [ID.ECN-Encap].

3. Benefit of using ECN to avoid congestion loss

When packet loss is a result of (mild) congestion, an ECN-enabled router may CE-mark, rather than drop an ECN-enabled packet. An application can benefit from this marking in several ways:

3.1. Improved Throughput

ECN can improve the throughput performance of an application, although this increase in throughput offered by ECN is often not the most significant gain.

When an application uses a light to moderately loaded network path, the number of packets that are dropped due to congestion is small. Using an example from Table 1 of [RFC3649], for a standard TCP sender with a Round Trip Time, RTT, of 0.1 seconds, a packet size of 1500 bytes and an average throughput of 1 Mbps, the average packet drop ratio is 0.02. This translates into an approximate 2% throughput gain if ECN is enabled. In heavy congestion, packet loss may be unavoidable with, or without, ECN [RFC2309.bis].

3.2. Reduced Head-of-Line Blocking

Many transports provide in-order delivery of received data segments to the applications they support. This requires that the transport stalls (or waits) for all data that was sent ahead of a particular segment to be correctly received before it can forward any later data. This is the usual requirement for TCP and SCTP. PR-SCTP [RFC3758], UDP, and DCCP [RFC4340] provide a transport that does not have this requirement.

Delaying data to provide in-order transmission to an application results in latency when segments are dropped as indications of congestion. The congestive loss creates a delay of at least one RTT for a loss event before data can be delivered to an application. We call this Head-of-Line (HOL) blocking.

In contrast, using ECN can remove the resulting delay following a loss that is a result of congestion:

- o First, the application receives the data normally - this also avoids dropping data that has already made it across the network

path. It avoids the additional delay of waiting for recovery of the lost segment when using a reliable transport.

- o Second, the transport receiver notes that it has received CE-marked packets, and then requests the sender to make an appropriate congestion-response to reduce the maximum transmission rate for future traffic.

3.3. Reduced Probability of RTO Expiry

In some situations, ECN can help reduce the chance of a retransmission timer expiring (e.g., expiry of the TCP or SCTP retransmission timeout, RTO [RFC5681]). When an application sends a burst of segments and then becomes idle (either because the application has no further data to send or the network prevents sending further data - e.g., flow or congestion control at the transport layer), the last segment of the burst may be lost. It is often not possible to recover this last segment (or last few segments) using standard methods such as Fast Recovery [RFC5681], since the receiver generates no feedback because it is unaware that the lost segments were actually sent.

In addition to avoiding HOL blocking, this allows the transport to avoid the consequent loss of state about the network path it is using, which would have arisen had there been a retransmission timeout. Typical impacts of a transport timeout are to reset path estimates such as the RTT, the congestion window, and possibly other transport state that can reduce the performance of the transport until it again adapts to the path.

Avoiding timeouts can hence improve the throughput of the application. This benefits applications that send intermittent bursts of data, and rely upon timer-based recovery of packet loss. It can be especially significant when ECN is used on TCP SYN/ACK packets as specified in [RFC5562] where the RTO interval may be large because in this case TCP cannot base the timeout period on prior RTT measurements from the same connection.

3.4. Applications that do not retransmit lost packets

Some latency-critical applications use transports that do not retransmit lost packets, yet these applications may be able to adjust the sending rate in the presence of congestion. Examples of such applications include UDP-based services that carry Voice over IP (VoIP), interactive video, or real-time data. The performance of many such applications degrades rapidly with increasing packet loss, and many therefore employ loss-hiding mechanisms (e.g., packet forward error correction, or data duplication) to mitigate the effect

of congestion loss on the application. However, such mechanisms add complexity and can themselves consume additional network capacity reducing the capacity for application data and contributing to the path latency when congestion is experienced.

By decoupling congestion control from loss, ECN can allow the transports supporting these applications to reduce their rate before the application experiences loss from congestion, especially when the congestion is mild and the application/transport can react promptly to reception of a CE-marked packet. Because this reduces the negative impact of using loss-hiding mechanisms, ECN can have a direct positive impact on the quality experienced by the users of these applications.

4. Benefit from Early Congestion Detection

An application can further benefit from using ECN, when the network devices are configured such that they mark packets at a lower level of congestion before they would otherwise have dropped packets from queue overflow:

4.1. Avoiding Capacity Overshoot

Internet transports do not know apriori how much capacity exists along a network path. Transports therefore try to measure the capacity available to an application by probing the network path with increasing traffic to the point where they detect the onset of congestion (such as TCP or SCTP Slow Start).

ECN can help capacity probing algorithms from significantly exceeding the bottleneck capacity of a network path. Since a transport that enables ECN can receive congestion signals before there is significant congestion, an early-marking method in network devices can help a transport respond before it induces significant congestion with resultant loss to itself or other applications sharing a common bottleneck. For example, an application/transport can avoid incurring significant congestion during Slow Start, or a bulk application that tries to increase its rate as fast as possible, may quickly detect the presence of congestion, causing it to promptly reduce its rate.

Use of ECN is more effective than transport mechanisms such as Limited Slow-Start [RFC3742] because it provides direct information about the state of the network path. An ECN-enabled application/transport that probes for capacity can reduce its rate as soon as it discovers CE-marked packets are received, and before the applications increases its rate to the point where it builds a queue in a network device that induces congestion loss. This benefits an application

seeking to increase its rate - but perhaps more significantly, it eliminates the often unwanted loss and queueing delay that otherwise may be inflicted on flows that share a common bottleneck.

4.2. Making Congestion Visible

A characteristic of using ECN is that it exposes the presence of congestion on a network path to the transport and network layers. This information could be used for monitoring the performance of the path, and could be used to directly meter the amount of congestion that has been encountered upstream on a path; metering packet loss is harder. This is used by Congestion Exposure (CoNex) [RFC6789].

A network flow that only experiences CE-marks and no loss implies that the sending endpoint is experiencing only congestion and not other sources of packet loss (e.g., link corruption or loss in middleboxes). The converse is not true - a flow may experience a mixture of ECN-marks and loss when there is only congestion or when there is a combination of packet loss and congestion [RFC2309.bis]. Recording the presence of CE-marked packets can therefore provide information about the performance of the network path.

5. Other forms of ECN-Marking/Reactions

The ECN mechanism defines both how packets are CE-marked and how transports need to react to reception of marked packets. This section describes the benefits when updated methods are used.

Benefit has been noted when packets are CE-marked earlier than they would otherwise be dropped, using an instantaneous queue, and if the receiver provides precise feedback about the number of packet marks encountered, a better sender behavior is possible. This has been shown by Datacenter TCP (DCTCP) [AL10].

Precise feedback about the number of packet marks encountered is supported by RTP over UDP [RFC6679] and proposed for SCTP [ST14] and TCP [KU13]. An underlying assumption of DCTCP is that it is deployed in confined environments such as a datacenter. It is currently unknown whether or how such behaviour could be introduced into the Internet.

6. Conclusion

Network devices should enable ECN and people configuring host stacks should also enable ECN. These are pre-requisites to allow applications to gain the benefits of ECN.

Application developers should where possible use transports that enable the benefits of ECN. Applications that directly use UDP need to provide support to implement the functions required for ECN. Once enabled, an application that uses a transport that supports ECN will experience the benefits of ECN as network deployment starts to enable ECN. The application does not need to be rewritten to gain these benefits.

Table 1 summarizes some of these benefits.

Section	Benefit
2.1	Improved Throughput
2.2	Reduced Head-of-Line
2.3	Reduced Probability of RTO Expiry
2.4	Applications that do not retransmit lost packets
3.1	Avoiding Capacity Overshoot
3.2	Making Congestion Visible

Table 1: Summary of Key Benefits from using ECN

7. Acknowledgements

The authors were part-funded by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700). The views expressed are solely those of the authors.

8. IANA Considerations

XXRFC ED - PLEASE REMOVE THIS SECTION XXX

This memo includes no request to IANA.

9. Security Considerations

This document introduces no new security considerations. Each RFC listed in this document discusses the security considerations of the specification it contains.

10. Revision Information

RFC-Ed please remove this section prior to publication.

Revision 00 was the first WG draft.

Revision 01 includes updates to complete sections and improve readability. Added section 2.

Comments are welcome to the authors or via the IETF AQM or TSVWG mailing lists.

11. References

11.1. Normative References

- [RFC2309.bis] Baker, F. and G. Fairhurst, "IETF Recommendations Regarding Active Queue Management", Internet-draft draft-ietf-aqm-recommendation-06, October 2014.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

11.2. Informative References

- [AL10] Alizadeh, M., Greenberg, A., Maltz, D., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S., and M. Sridharan, "Data Center TCP (DCTCP)", SIGCOMM 2010, August 2010.
- [ID.ECN-Encap] Briscoe, B., Kaippallimalil, J., and P. Thaler, "Guidelines for Adding Congestion Notification to Protocols that Encapsulate IP , IETF work-in-progress: draft-ietf-tsvwg-ecn-encap-guidelines", .
- [KH13] Khademi, N., Ros, D., and M. Welzl, "The New AQM Kids on the Block: Much Ado About Nothing?", University of Oslo Department of Informatics technical report 434, October 2013.
- [KU13] Kuehlewind, M. and R. Scheffenegger, "Problem Statement and Requirements for a More Accurate ECN Feedback", Internet-draft draft-ietf-tcpm-accecn-reqs-04.txt, October 2013.
- [RFC2884] Hadi Salim, J. and U. Ahmed, "Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks", RFC 2884, July 2000.
- [RFC3649] Floyd, S., "HighSpeed TCP for Large Congestion Windows", RFC 3649, December 2003.

- [RFC3742] Floyd, S., "Limited Slow-Start for TCP with Large Congestion Windows", RFC 3742, March 2004.
- [RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, May 2004.
- [RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, March 2006.
- [RFC5405] Eggert, Lars. and Gorry. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", November 2008.
- [RFC5562] Kuzmanovic, A., Mondal, A., Floyd, S., and K. Ramakrishnan, "Adding Explicit Congestion Notification (ECN) Capability to TCP's SYN/ACK Packets", RFC 5562, June 2009.
- [RFC5681] "TCP Congestion Control", .
- [RFC6040] "Tunnelling of Explicit Congestion Notification", .
- [RFC6679] Westerlund, M., Johansson, I., Perkins, C., O'Hanlon, P., and K. Carlberg, "Explicit Congestion Notification (ECN) for RTP over UDP", RFC 6679, August 2012.
- [RFC6789] Briscoe, B., Woundy, R., and A. Cooper, "Congestion Exposure (ConEx) Concepts and Use Cases", RFC 6789, December 2012.
- [ST14] Stewart, R., Tuexen, M., and X. Dong, "ECN for Stream Control Transmission Protocol (SCTP)", Internet-draft draft-stewart-tsvwg-sctpecn-05.txt, January 2014.

Authors' Addresses

Michael Welzl
University of Oslo
PO Box 1080 Blindern
Oslo N-0316
Norway

Phone: +47 22 85 24 20
Email: michawe@ifi.uio.no

Godred Fairhurst
University of Aberdeen
School of Engineering, Fraser Noble Building
Aberdeen AB24 3UE
UK

Email: gorry@erg.abdn.ac.uk

Active Queue Management and Packet Scheduling (aqm)
Internet-Draft
Intended status: Informational
Expires: July 16, 2015

G. White
CableLabs
R. Pan
Cisco Systems
January 12, 2015

A PIE-Based AQM for DOCSIS Cable Modems
draft-white-aqm-docsis-pie-02

Abstract

DOCSIS cable modems provide broadband Internet access to over one hundred million users worldwide. They are commonly positioned at the head of the bottleneck link for traffic in the upstream direction (from the customer), and as a result, the impact of buffering and bufferbloat in the cable modem can have a significant effect on user experience. The CableLabs DOCSIS 3.1 specification includes requirements for cable modems to support an Active Queue Management (AQM) algorithm that is intended to alleviate the impact that buffering has on latency sensitive traffic, while preserving bulk throughput performance. In addition, the CableLabs DOCSIS 3.0 specifications have also been amended to contain similar requirements.

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 <http://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 July 16, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Overview of DOCSIS AQM Requirements	2
2. The DOCSIS MAC Layer and Service Flows	3
3. DOCSIS-PIE vs. PIE	4
3.1. Latency Target	4
3.2. Departure rate estimation	5
3.3. Expanded auto-tuning range	6
3.4. Trigger for exponential decay	6
4. Implementation Guidance	6
5. References	7
Appendix A. DOCSIS-PIE Algorithm definition	7
A.1. DOCSIS-PIE AQM Constants and Variables	7
A.1.1. Configuration parameters	7
A.1.2. Constant values	8
A.1.3. Variables	8
A.1.4. Public/system functions:	9
A.2. DOCSIS-PIE AQM Control Path	9
A.3. DOCSIS-PIE AQM Data Path	11
Appendix B. Change Log	13
B.1. Since draft-white-aqm-docsis-pie-01	13
Authors' Addresses	13

1. Overview of DOCSIS AQM Requirements

CableLabs' DOCSIS 3.1 specification [DOCSIS_3.1] mandates that cable modems implement a specific variant of the Proportional Integral controller Enhanced (PIE) [I-D.ietf-aqm-pie] active queue management algorithm. This specific variant is provided for reference in Appendix A. CableLabs' DOCSIS 3.0 specification [DOCSIS_3.0] has been amended to recommend that cable modems implement the same algorithm. Both specifications allow that cable modems can optionally implement additional algorithms, that can then be selected for use by the operator via the modem's configuration file.

These requirements on the cable modem apply to upstream transmissions.

Both specifications also include requirements (mandatory in DOCSIS 3.1 and recommended in DOCSIS 3.0) that the Cable Modem Termination System (CMTS) implement active queue management for downstream traffic, however no specific algorithm is defined for downstream use.

2. The DOCSIS MAC Layer and Service Flows

The DOCSIS Media Access Control (sub-)layer provides tools for configuring differentiated Quality of Service for different applications by the use of Packet Classifiers and Service Flows.

Each cable modem can be configured with multiple Packet Classifiers and Service Flows. The maximum number of such entities that a cable modem supports is an implementation decision for the manufacturer, but modems typically support 16 or 32 Service Flows and at least that many Packet Classifiers.

Each Service Flow has an associated Quality of Service (QoS) parameter set that defines the treatment of the packets that traverse the Service Flow. These parameters include (for example) Minimum Reserved Traffic Rate, Maximum Sustained Traffic Rate, Peak Traffic Rate, Maximum Traffic Burst, Traffic Priority. Each upstream Service Flow corresponds to a queue in the cable modem, and each downstream Service Flow corresponds to a queue in the CMTS. The DOCSIS AQM requirements mandate that the CM and CMTS implement the AQM algorithm (and allow it to be disabled if need be) on each Service Flow queue independently.

Packet Classifiers can match packets based upon several fields in the packet/frame headers including the Ethernet header, IP header, and TCP/UDP header. Matched packets are then queued in the associated Service Flow queue.

It is typical that upstream and downstream Service Flows used for broadband Internet access are configured with a Maximum Sustained Traffic Rate. This QoS parameter rate-shapes the traffic onto the DOCSIS link, and is the main parameter that defines the service offering. Additionally, it is common that upstream and downstream Service Flows are configured with a Maximum Traffic Burst and a Peak Traffic Rate. These parameters allow the service to burst at a higher (sometimes significantly higher) rate than is defined in the Maximum Sustained Traffic Rate for the amount of bytes configured in Maximum Traffic Burst, as long as the long-term average data rate remains at or below the Maximum Sustained Traffic Rate.

Mathematically, what is enforced is that the traffic placed on the DOCSIS link in the time interval $(t1, t2)$ complies with the following rate shaping equations:

$$\text{TxBytes}(t1,t2) \leq (t2-t1)*R/8 + B$$

$$\text{TxBytes}(t1,t2) \leq (t2-t1)*P/8 + 1522$$

for all values $t2 > t1$, where:

R = Maximum Sustained Traffic Rate (bps)

P = Peak Traffic Rate (bps)

B = Maximum Traffic Burst (bytes)

The result of this configuration is that the link rate available to the Service Flow varies based on the pattern of load. If the load that the Service Flow places on the link is less than the Maximum Sustained Traffic Rate, the Service Flow "earns" credit that it can then use (should the load increase) to burst at the Peak Traffic Rate. This dynamic is important since these rate changes (particularly the decrease in data rate once the traffic burst credit is exhausted) can induce a step increase in buffering latency.

3. DOCSIS-PIE vs. PIE

There are a number of differences between the version of the PIE algorithm that is mandated for cable modems in the DOCSIS specifications and the version described in [I-D.ietf-aqm-pie].

- o 10 ms default latency target, configurable per service flow
- o departure rate estimation
- o expanded auto-tuning range
- o trigger for exponential decay

3.1. Latency Target

The latency target (aka delay reference) is a key parameter that affects, among other things, the tradeoff in performance between latency-sensitive applications and bulk TCP applications. Via simulation studies, a value of 10ms was identified as providing a good balance of performance. However, it is recognized that there may be service offerings for which this value doesn't provide the best performance balance. As a result, this is provided as a configuration parameter that the operator can set independently on each upstream service flow. If not explicitly set by the operator, the modem will use 10 ms as the default value.

3.2. Departure rate estimation

The PIE algorithm utilizes a departure rate estimator to track fluctuations in the egress rate for the queue and to generate a smoothed estimate of this rate for use in the drop probability calculation. This estimator may be well suited to many link technologies, but is not ideal for DOCSIS upstream links for a number of reasons.

First, the bursty nature of the upstream transmissions, in which the queue drains at line rate (up to ~100 Mbps for DOCSIS 3.0 and ~1 Gbps for DOCSIS 3.1) and then is blocked until the next transmit opportunity, results in the potential for inaccuracy in measurement, given that the PIE departure rate estimator starts each measurement during a transmission burst and ends each measurement during a (possibly different) transmission burst. For example, in the case where the start and end of measurement occur within a single burst, the PIE estimator will calculate the egress rate to be equal to the line rate, rather than the average rate available to the modem.

Second, the latency introduced by the DOCSIS request-grant mechanism can result in some further inaccuracy. In typical conditions, the request-grant mechanism can add between ~4 ms and ~8 ms of latency to the forwarding of upstream traffic. Within that range, the amount of additional latency that affects any individual data burst is effectively random, being influenced by the arrival time of the burst relative to the next request transmit opportunity, among other factors.

Third, in the significant majority of cases, the departure rate, while variable, is controlled by the modem itself via the pair of token bucket rate shaping equations described in Section 2. Together, these two equations enforce a maximum sustained traffic rate, a peak traffic rate, and a maximum traffic burst size for the modem's requested bandwidth. The implication of this is that the modem, in the significant majority of cases, will know precisely what the departure rate will be, and can predict exactly when transitions between peak rate and maximum sustained traffic rate will occur. Compare this to the PIE estimator, which would be simply reacting to (and smoothing its estimate of) those rate transitions after the fact.

Finally, since the modem is already implementing the dual token bucket traffic shaper, it contains enough internal state to calculate predicted queuing delay with a minimum of computations. Furthermore, these computations only need to be run every drop probability update interval, as opposed to the PIE estimator, which runs a similar number of computations on each packet dequeue event.

For these reasons, the DOCSIS-PIE algorithm utilizes the configuration and state of the dual token bucket traffic shaper to translate queue depth into predicted queuing delay, rather than implementing the departure rate estimator defined in PIE.

3.3. Expanded auto-tuning range

The PIE algorithm scales the PI coefficients based on the current drop probability. The DOCSIS-PIE algorithm extends this scaling to drop probabilities below $1e-4$.

3.4. Trigger for exponential decay

The PIE algorithm includes a mechanism by which the drop probability is allowed to decay exponentially (rather than linearly) when it is detected that the buffer is empty. In the DOCSIS case, recently arrived packets may reside in buffer due to the request-grant latency even if the link is effectively idle. As a result, the buffer may not be identically empty in the situations for which the exponential decay is intended. To compensate for this, we trigger exponential decay when the buffer occupancy is less than $5\text{ms} * \text{Peak Traffic Rate}$.

4. Implementation Guidance

The AQM space is an evolving one, and it is expected that continued research in this field may in the future result in improved algorithms.

As part of defining the DOCSIS-PIE algorithm, we split the pseudocode definition into two components, a "data path" component and a "control path" component. The control path component contains the packet drop probability update functionality, whereas the data path component contains the per-packet operations, including the drop decision logic.

It is understood that some aspects of the cable modem implementation may be done in hardware, particularly functions that handle packet-processing.

While the DOCSIS specifications don't mandate the internal implementation details of the cable modem, modem implementers are strongly advised against implementing the control path functionality in hardware. The intent of this advice is to retain the possibility that future improvements in AQM algorithms can be accommodated via software updates to deployed devices.

5. References

[DOCSIS_3.0]

CableLabs, "DOCSIS 3.0 MAC and Upper Layer Protocols Specification", November 2013, <<http://www.cablelabs.com/wp-content/uploads/specdocs/CM-SP-MULPIv3.0-I23-131120.pdf>>.

[DOCSIS_3.1]

CableLabs, "DOCSIS 3.1 MAC and Upper Layer Protocols Specification", October 2013, <<http://www.cablelabs.com/wp-content/uploads/specdocs/CM-SP-MULPIv3.1-I01-131029.pdf>>.

[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-00 (work in progress), October 2014.

Appendix A. DOCSIS-PIE Algorithm definition

PIE defines two functions organized here into two design blocks:

1. Control path block, a periodically running algorithm that calculates a drop probability based on the estimated queuing latency and queuing latency trend.
2. Data path block, a function that occurs on each packet enqueue: per-packet drop decision based on the drop probability.

It is desired to have the ability to update the Control path block based on operational experience with PIE deployments.

A.1. DOCSIS-PIE AQM Constants and Variables

A.1.1. Configuration parameters

- o LATENCY_TARGET. AQM Latency Target for this Service Flow
- o PEAK_RATE. Service Flow configured Peak Traffic Rate, expressed in Bytes/sec.
- o MSR. Service Flow configured Max. Sustained Traffic Rate, expressed in Bytes/sec.
- o BUFFER_SIZE. The size (in bytes) of the buffer for this Service Flow.

A.1.2. Constant values

- o A=0.25, B=2.5. Weights in the drop probability calculation
- o INTERVAL=16 ms. Update interval for drop probability.
- o DELAY_HIGH=200 ms.
- o BURST_RESET_TIMEOUT = 1 s.
- o MAX_BURST = 142 ms (150 ms-8 ms(update error))
- o MEAN_PKTSIZE = 1024 bytes
- o MIN_PKTSIZE = 64 bytes
- o PROB_LOW = 0.85
- o PROB_HIGH = 8.5
- o LATENCY_LOW = 5 ms

A.1.3. Variables

- o drop_prob_. The current packet drop probability.
- o accu_prob_. accumulated drop prob. since last drop
- o qdelay_old_. The previous queue delay estimate.
- o burst_allowance_. Countdown for burst protection, initialize to 0
- o burst_reset_. counter to reset burst
- o burst_state_. Burst protection state encoding 3 states:
 - NOBURST - no burst yet
 - FIRST_BURST - first burst detected, no protection yet
 - PROTECT_BURST - first burst detected, protecting burst if burst_allowance_ > 0
- o queue_. Holds the pending packets.

A.1.4. Public/system functions:

- o `drop(packet)`. Drops/discards a packet
- o `random()`. Returns a uniform r.v. in the range 0 ~ 1
- o `queue_.is_full()`. Returns true if `queue_` is full
- o `queue_.byte_length()`. Returns current `queue_` length in bytes, including all MAC PDU bytes without DOCSIS MAC overhead
- o `queue_.enqueue(packet)`. Adds packet to tail of `queue_`
- o `msrtokens()`. Returns current token credits (in bytes) from the Max Sust. Traffic Rate token bucket
- o `packet.size()`. Returns size of packet

A.2. DOCSIS-PIE AQM Control Path

The DOCSIS-PIE control path performs the following:

- o Calls `control_path_init()` at service flow creation
- o Calls `calculate_drop_prob()` at a regular INTERVAL (16ms)

```
=====
// Initialization function
control_path_init() {
    drop_prob_ = 0;
    qdelay_old_ = 0;
    burst_reset_ = 0;
    burst_state_ = NOBURST;
}

// Background update, occurs every INTERVAL
calculate_drop_prob() {

    if (queue_.byte_length() <= msrtokens()) {
        qdelay = queue_.byte_length() / PEAK_RATE;
    } else {
        qdelay = ((queue_.byte_length() - msrtokens()) / MSR \
                  + msrtokens() / PEAK_RATE);
    }

    if (burst_allowance_ > 0) {
        drop_prob_ = 0;
    } else {
```

```
p = A * (qdelay - LATENCY_TARGET) + \
    B * (qdelay - qdelay_old_);
// Since A=0.25 & B=2.5, can be implemented
// with shift and add

if (drop_prob_ < 0.000001) {
    p /= 2048;
} else if (drop_prob_ < 0.00001) {
    p /= 512;
} else if (drop_prob_ < 0.0001) {
    p /= 128;
} else if (drop_prob_ < 0.001) {
    p /= 32;
} else if (drop_prob_ < 0.01) {
    p /= 8;
} else if (drop_prob_ < 0.1) {
    p /= 2;
} else if (drop_prob_ < 1) {
    p /= 0.5;
} else if (drop_prob_ < 10) {
    p /= 0.125;
} else {
    p /= 0.03125;
}

if ((drop_prob_ >= 0.1) && (p > 0.02)) {
    p = 0.02;
}
drop_prob_ += p;

/* for non-linear drop in prob */
if (qdelay < LATENCY_LOW && qdelay_old_ < LATENCY_LOW) {
    drop_prob_ *= 0.98; // (1-1/64) is sufficient
} else if (qdelay > DELAY_HIGH) {
    drop_prob_ += 0.02;
}

drop_prob_ = max(0, drop_prob_);
drop_prob_ = min(drop_prob_, \
    PROB_LOW * MEAN_PKT_SIZE/MIN_PKT_SIZE);
}

if (burst_allowance_ < INTERVAL)
    burst_allowance_ = 0;
else
    burst_allowance_ = burst_allowance_ - INTERVAL;

// both old and new qdelay is well better than the
```

```

// target and drop_prob_ == 0, time to clear burst tolerance
if ((qdelay < 0.5 * LATENCY_TARGET)
    && (qdelay_old_ < 0.5 * LATENCY_TARGET)
    && (drop_prob_ == 0)
    && (burst_allowance_ == 0)){

    if (burst_state_ == PROTECT_BURST) {
        burst_state_ = FIRST_BURST;
        burst_reset_ = 0;

    } else if (burst_state_ == FIRST_BURST) {
        burst_reset_ += INTERVAL ;
        if (burst_reset_ > BURST_RESET_TIMEOUT) {
            burst_reset_ = 0;
            burst_state_ = NOBURST;
        }
    }
    } else if (burst_state_ == FIRST_BURST) {
        burst_reset_ = 0;
    }

    qdelay_old_ = qdelay;
}

```

A.3. DOCSIS-PIE AQM Data Path

The DOCSIS-PIE data path performs the following:

- o Calls enqueue() in response to an incoming packet from the CMCI

```

=====
enqueue(packet) {

    if (queue_.is_full()) {
        drop(packet);
        accu_prob_ = 0;
    } else if (drop_early(packet, queue_.byte_length())) {
        drop(packet);
    } else {
        queue_.enqueue(packet);
    }
}

//////////
drop_early(packet, queue_length) {
    if (burst_allowance_ > 0) {
        return FALSE;
    }
}

```

```
}

if (drop_prob_ == 0) {
    accu_prob_ = 0;
}

if (burst_state_ == NOBURST) { //first burst?
    if (queue_.byte_length() < BUFFER_SIZE/3) {
        return FALSE;
    } else {
        burst_state_ = FIRST_BURST; //burst detected
    }
}

//The CM can quantize packet.size to 64, 128, 256, 512, 768,
// 1024, 1280, 1536, 2048 in the calculation below
p1 = drop_prob_ * packet.size() / MEAN_PKTSIZE;
p1 = min(p1, PROB_LOW);

accu_prob_ += p1;

// If latency is low, don't drop packets
if ( (qdelay_old_ < 0.5 * LATENCY_TARGET && drop_prob_ < 0.2)
    || (queue_.byte_length() <= 2 * MEAN_PKTSIZE) ) {
    return FALSE;
}

drop = TRUE;
if (accu_prob_ < PROB_LOW) { // avoid dropping too fast due
    drop = FALSE;           // to bad luck of coin tosses...
} else if (accu_prob_ >= PROB_HIGH) { // ...and avoid droppping
    drop = TRUE;             // too slowly
} else { //Random drop
    double u = random(); // 0 ~ 1
    if (u > p1) {
        drop = FALSE;
    }
}

if (drop == FALSE) return FALSE;

// In case of packet drop:
accu_prob_ = 0;

// Not protecting burst yet? Start protecting burst.
// This will set the burst_allowance_ value, and
// calculate_drop_prob() will decrement it.
// Could implement this as a 150ms timer instead.
```



```
    if (burst_state_ == FIRST_BURST) {  
        burst_state_ = PROTECT_BURST;  
        burst_allowance_ = MAX_BURST;  
    }  
    return TRUE;  
}
```

Appendix B. Change Log

B.1. Since draft-white-aqm-docsis-pie-01

Added Change Log.

Removed discussion of Packet drop de-randomization, Enhanced burst protection, and 16ms update interval, as these are now included in [I-D.ietf-aqm-pie].

Authors' Addresses

Greg White
CableLabs
858 Coal Creek Circle
Louisville, CO 80027-9750
USA

Email: g.white@cablelabs.com

Rong Pan
Cisco Systems
510 McCarthy Blvd
Milpitas, CA 95134
USA

Email: ropan@cisco.com