Network Working Group F. Baker, Ed.
Internet-Draft Cisco Systems
Obsoletes: 2309 (if approved) G. Fairhurst, Ed.
Intended status: Best Current Practice University of Aberdeen
Expires: January 10, 2014 July 09, 2013

IETF Recommendations Regarding Active Queue Management
draft-baker-aqm-recommendation-02

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 largely repeats the recommendations of RFC 2309, updated after fifteen years of experience and new research.

Status of This Memo

Copyright Notice

Table of Contents

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 of the mid 1980s [RFC0896][RFC0970], and is technically called "congestive collapse".

The original fix for Internet meltdown was provided by Van Jacobsen. Beginning in 1986, Jacobsen developed the congestion avoidance mechanisms that are now required in TCP implementations [Jacobson88] [RFC1122]. 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., marked or dropped packets) from the network. It is primarily these TCP congestion avoidance algorithms that prevent the congestive collapse of today's Internet.

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 TCP 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 the network devices to complement the endpoint congestion avoidance mechanisms. These mechanisms may be implemented in network devices that include routers, switches, and other network middleboxes.

It is useful to distinguish between two classes of algorithms related to congestion control: "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 AQM mechanisms are closely related, they address different performance issues.

This memo highlights two performance issues:

The first issue is the need for an advanced form of queue management that we call "active queue management." Section 2 summarizes the benefits that active queue management can bring. A number of Active Queue Management (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 3 of this memo, is the potential for future congestive collapse of the Internet due to flows that are unresponsive, or not sufficiently responsive, to congestion indications. Unfortunately, there is no consensus solution to controlling congestion caused by such aggressive flows; significant research and engineering will be required before any solution will be available. It is imperative that this work be energetically pursued, to ensure the future stability of the Internet.

Section 4 concludes the memo with a set of recommendations to the Internet community concerning these topics.

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 RTT or a small number of RTTs, for elastic traffic [RFC1633].

[RFC2309] 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). This update results from experience with this and other algorithms, and the Active Queue Management discussion within the IETF.

1.1. 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

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 two important drawbacks.

1. Lock-Out

   In some situations tail drop allows a single connection or a few flows to monopolize queue space, preventing other connections from getting room in the queue. This "lock-out" phenomenon is often the result of synchronization or other timing effects.

2. 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. 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. This can result in a global synchronization of flows throttling back, followed by a sustained period of lowered link utilization, reducing overall throughput.

   The point 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 the transmission of bursty data. Normally we would like to have mall queues 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 short, 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.

Besides tail drop, two alternative queue disciplines that can be applied when a queue becomes full are "random drop on full" or "drop front on full". Under the random drop on full discipline, a 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 the queue is full and a new packet

arrives. Under the "drop front on full" discipline [Lakshman96], the network device drops the packet at the front of the queue when the queue is full and a new packet arrives. 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 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 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 possible 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, Telnet traffic, 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. For example, a router may maintain per-flow state to achieve general fairness by a per-flow scheduling algorithm such as Fair Queueing (FQ) [Demers90], or a Class-Based Queue scheduling algorithm such as CBQ [Floyd95].

In contrast, AQM is needed even for network devices that use per-flow scheduling algorithms such as FQ or class-based scheduling algorithms, such as CBQ. This is because per-flow scheduling algorithms by themselves do not control the overall queue size or the size of individual queues. AQM is needed to control the overall average queue sizes, so that arriving bursts can be accommodated without dropping packets. In addition, AQM should be used to control the queue size for each individual flow or class, so that they do not experience unnecessarily high delay. Therefore, AQM should be applied across the classes or flows as well as within each class or flow.

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

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].

We call a flow "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.

It is convenient to divide flows into three classes: (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 not TCP-friendly. The last two classes contain more aggressive flows that pose significant threats to Internet performance, which we will now discuss.

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, MTU, etc.). This is described in the remainder of the document.

2. Non-Responsive Flows

   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].

   There is a growing set of UDP-based applications whose congestion avoidance algorithms are inadequate or nonexistent (i.e, a flow that does not throttle its sending rate when it experiences congestion). Examples include some UDP streaming applications for packet voice and video, and some multicast bulk data transport. If no action is taken, such unresponsive flows could lead to a new congestive collapse [RFC2309].

In general, UDP-based applications need to incorporate effective congestion avoidance mechanisms [RFC5405]. Further research and development of ways to accomplish congestion avoidance for presently unresponsive applications continue to be important.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, DCCP) or by incorporating their own congestion control in the application. [RFC5405].

3. Non-TCP-friendly Transport Protocols

A second threat is posed by transport protocol implementations that are responsive to congestion, but, either deliberately or through faulty implementation, are not TCP-friendly. Such applications 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 over responds over an excessively long time period. Such flows are unlikely to be responsive to congestion signals in a time frame 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 of multiple (short) TCP flows can be more aggressive than standard bulk TCP. Applications (e.g. web browsers and peer-to-peer file-sharing) have exploited this by opening multiple connections to the same endpoint.

The projected increase in the fraction of total Internet traffic for more aggressive flows in classes 2 and 3 clearly poses a threat to

future Internet stability. There is 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 identifying and isolating unresponsive or non-TCP-friendly flows at an acceptable overhead cost. 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, DSCP); 2) a source/destination host pair (IP addresses, DSCP); 3) a given source host or a given destination host. We suggest that the source/destination host pair gives the most appropriate granularity in many circumstances. However, it is possible that different vendors/providers could set different granularities for defining a flow (as a way of "distinguishing" themselves from one another), or that different granularities could 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. This document updates these to include:

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. The algorithms that the IETF recommends SHOULD NOT require operational (especially manual) configuration or tuning.

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. However, 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 on. To the contrary, available research [Papagiannaki] says that even high speed links in network cores that are normally very stable in depth and behavior experience occasional issues that need moderation.

4.1. Operational deployments SHOULD use AQM procedures

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

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:

- Delaying data segments in flight, such as in a queue.

- Dropping traffic in transit.

- Marking traffic, such as using Explicit Congestion Control[RFC3168] [RFC4301] [RFC4774] [RFC6040] [RFC6679].

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].

Increased network latency can be 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 RTP, this impacts the RTCP-reported RTT and can trigger a sender to adjust its rate. For example, LEDBAT [RFC6817] assumes delay as a primary signal of congestion.

It is essential that all Internet hosts respond to loss [RFC5681], [RFC5405][RFC2960][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 transport must the latter." Unreliable transports (e.g. using UDP) need to similarly react to loss [RFC5405]

Network devices SHOULD use use an AQM algorithm to determine which packets are effected by congestion.

Loss 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 retransmitted. While other applications/transports may adapt to the absence of the data, this still implies inefficient use of available capacity and the dropped traffic can affect other flows. Hence, 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 and functions implemented in network devices, the latter rely upon using AQM.

Congestion for ECN-capable transports is instead 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 necessary 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 devices may similarly "hide" and ECN mark. In normal operation such cases should be very uncommon.

Network devices SHOULD use an algorithm to drop excessive traffic, even when marked as originating from an ECN capable transport.

4.3. AQM algorithms deployed SHOULD NOT require operational tuning

A number of algorithms have been proposed. Many require some form of tuning or initial condition. This can make them difficult to use operationally. Hence, self-tuning algorithms are to be preferred. The algorithms that the IETF recommends SHOULD 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 characterised by particular profiles of packet size this should not be used as the basis for AQM. 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 actual or projected time 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 [Byte-pkt].

Applications (or transports) generally know what packet size they are using and can hence make their judgements 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 [Byte-pkt].

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 is 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. AQM algorithms also need to consider use of tunnel encapsulations, which 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.

When speaking of TCP performance, the terms "knee" and "cliff" area defined by [Jain94]. They respectively refer to the minimum congestion window that maximises throughput and the maximum congestion window that avoids loss. An application that transmits at the rate determined by this window has the effect of maximizing the rate or throughput. For the sender, exceeding the cliff is ineffective, as it (by definition) induces loss; operating at a point close to the cliff has a negative impact on other traffic and applications, triggering operator activities, such as those discussed in [RFC6057]. Operating below the knee reduces the throughput, since the sender fails to use available network capacity.

If the objective is to deliver data from its source to its recipient in the least possible time, as a result, the behavior of any elastic transport congestion control algorithm should seek to use an effective window at or above the knee and well below the cliff. Choice of an appropriate rate can significantly impact the loss and delay experienced not only by a flow but by other flows that share the same 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, gaming, transaction-based protocols). Such applications may not wish to maximise throughput, but may also desire a lower loss rate or bounded delay.

Transport protocols and applications need timely signals of congestion. The time taken to detect and respond to congestion is assisted by network devices not dropping long runs of packets from the same flow. It is difficult to detect tail losses at a higher layer and may sometimes require timers or probes to detect and respond to such loss.

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 in 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 be found that offer significant advantages for deployed networks. There is also renewed interest in deploying AQM and the potential of ECN.

An obvious example of further research in 2013 is the need to consider the use of Map/Reduce applications in data centers; do we need to extend our taxonomy of TCP/SCTP sessions to include not only "mice" and "elephants", but "lemmings"? "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 data center applications.

Examples of other required research include:

- o Research into new AQM and scheduling algorithms.

- o Research into the use of and 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.

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. We note, however, that denial-of-service attacks may create unresponsive traffic flows that may be indistinguishable from other flows (e.g. tunnels carrying aggregates of short flows, high-rate isochronous applications). This threat exists also in network devices that do not deploy AQM, but when AQM is deployed could be used to degrade the benefit of the new method. The recommendations support ongoing research applicable to such attacks.

7. Privacy Considerations

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

8. Acknowledgements

The original recommendation in [RFC2309] 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. This is an edited version of that document, with much of its text and arguments unchanged.

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 came from Colin Perkins, Richard Scheffenegger, and Dave Taht.

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

[Byte-pkt]
Internet Engineering Task Force, Work in Progress, "Byte and Packet Congestion Notification (draft-ietf-tsvwg-byte-pkt-congest)", July 2013.

[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.

9.2. Informative References

[Demers90]
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.

[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.

[Papagiannaki] Sprint ATL, KAIST, University of Minnesota, Sprint ATL, Intel Research, "Analysis of Point-To-Point Packet Delay In an Operational Network", IEEE Infocom 2004, March 2004, <http://www.ieee-infocom.org/2004/Papers/37_4.PDF>.

[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.

[RFC2960] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L., and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, October 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.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.

[RFC6057] Bastian, C., Klieber, T., Livingood, J., Mills, J., and R. Woundy, "Comcast's Protocol-Agnostic Congestion Management System", RFC 6057, December 2010.

[RFC6817] Shalunov, S., Hazel, G., Iyengar, J., and M. Kuehlewind, "Low Extra Delay Background Transport (LEDBAT)", RFC 6817, December 2012.

[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.

Appendix A. Change Log

Initial Version: March 2013

Minor uphe algorithms that the IETF recommends SHOULD NOT require operational (especially manual) configuration or tuningdate:
April 2013

-02; Major surgery. This draft is for discussion at IETF-87 and expected to be further updated.
July 2013

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