Internet Engineering Task Force (IETF)
Request for Comments: 7141
BCP: 41
Updates: 2309, 2914
Category: Best Current Practice
ISSN: 2070-1721

B. Briscoe
BT
J. Manner
Aalto University

February 2014

Byte and Packet Congestion Notification

Abstract

This document provides recommendations of best current practice for dropping or marking packets using any active queue management (AQM) algorithm, including Random Early Detection (RED), BLUE, Pre-Congestion Notification (PCN), and newer schemes such as CoDel (Controlled Delay) and PIE (Proportional Integral controller Enhanced). We give three strong recommendations: (1) packet size should be taken into account when transports detect and respond to congestion indications, (2) packet size should not be taken into account when network equipment creates congestion signals (marking, dropping), and therefore (3) in the specific case of RED, the byte-mode packet drop variant that drops fewer small packets should not be used. This memo updates RFC 2309 to deprecate deliberate preferential treatment of small packets in AQM algorithms.

Status of This Memo

This memo documents an Internet Best Current Practice.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on BCPs is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7141.

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

This document provides recommendations of best current practice for how we should correctly scale congestion control functions with respect to packet size for the long term. It also recognises that expediency may be necessary to deal with existing widely deployed protocols that don't live up to the long-term goal.

When signalling congestion, the problem of how (and whether) to take packet sizes into account has exercised the minds of researchers and practitioners for as long as active queue management (AQM) has been discussed. Indeed, one reason AQM was originally introduced was to reduce the lock-out effects that small packets can have on large packets in tail-drop queues. This memo aims to state the principles we should be using and to outline how these principles will affect future protocol design, taking into account pre-existing deployments.

The question of whether to take into account packet size arises at three stages in the congestion notification process:

Measuring congestion: When a congested resource measures locally how congested it is, should it measure its queue length in time, bytes, or packets?

Encoding congestion notification into the wire protocol: When a congested network resource signals its level of congestion, should the probability that it drops/marks each packet depend on the size of the particular packet in question?

Decoding congestion notification from the wire protocol: When a transport interprets the notification in order to decide how much to respond to congestion, should it take into account the size of each missing or marked packet?

Consensus has emerged over the years concerning the first stage, which Section 2.1 records in the RFC Series. In summary: If possible, it is best to measure congestion by time in the queue; otherwise, the choice between bytes and packets solely depends on whether the resource is congested by bytes or packets.

The controversy is mainly around the last two stages: whether to allow for the size of the specific packet notifying congestion i) when the network encodes or ii) when the transport decodes the congestion notification.

Currently, the RFC series is silent on this matter other than a paper trail of advice referenced from [RFC2309], which conditionally recommends byte-mode (packet-size dependent) drop [pktByteEmail].

Reducing the number of small packets dropped certainly has some tempting advantages: i) it drops fewer control packets, which tend to be small and ii) it makes TCP's bit rate less dependent on packet size. However, there are ways of addressing these issues at the transport layer, rather than reverse engineering network forwarding to fix the problems.

This memo updates [RFC2309] to deprecate deliberate preferential treatment of packets in AQM algorithms solely because of their size. It recommends that (1) packet size should be taken into account when transports detect and respond to congestion indications, (2) not when network equipment creates them. This memo also adds to the congestion control principles enumerated in BCP 41 [RFC2914].

In the particular case of Random Early Detection (RED), this means that the byte-mode packet drop variant should not be used to drop fewer small packets, because that creates a perverse incentive for transports to use tiny segments, consequently also opening up a DoS vulnerability. Fortunately, all the RED implementers who responded to our admittedly limited survey (Section 4.2.4) have not followed the earlier advice to use byte-mode drop, so the position this memo argues for seems to already exist in implementations.

However, at the transport layer, TCP congestion control is a widely deployed protocol that doesn't scale with packet size (i.e., its reduction in rate does not take into account the size of a lost packet). To date, this hasn't been a significant problem because most TCP implementations have been used with similar packet sizes. But, as we design new congestion control mechanisms, this memo recommends that we build in scaling with packet size rather than assuming that we should follow TCP's example.

This memo continues as follows. First, it discusses terminology and scoping. Section 2 gives concrete formal recommendations, followed by motivating arguments in Section 3. We then critically survey the advice given previously in the RFC Series and the research literature (Section 4), referring to an assessment of whether or not this advice has been followed in production networks (Appendix A). To wrap up, outstanding issues are discussed that will need resolution both to inform future protocol designs and to handle legacy AQM deployments (Section 5). Then security issues are collected together in Section 6 before conclusions are drawn in Section 7. The interested reader can find discussion of more detailed issues on the theme of byte vs. packet in the appendices.

This memo intentionally includes a non-negligible amount of material on the subject. For the busy reader, Section 2 summarises the recommendations for the Internet community.

1.1. Terminology and Scoping

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].

This memo applies to the design of all AQM algorithms, for example, Random Early Detection (RED) [RFC2309], BLUE [BLUE02], Pre-Congestion Notification (PCN) [RFC5670], Controlled Delay (CoDel) [CoDel], and the Proportional Integral controller Enhanced (PIE) [PIE]. Throughout, RED is used as a concrete example because it is a widely known and deployed AQM algorithm. There is no intention to imply that the advice is any less applicable to the other algorithms, nor that RED is preferred.

Congestion Notification: Congestion notification is a changing signal that aims to communicate the probability that the network resource(s) will not be able to forward the level of traffic load offered (or that there is an impending risk that they will not be able to).

The 'impending risk' qualifier is added, because AQM systems set a virtual limit smaller than the actual limit to the resource, then notify the transport when this virtual limit is exceeded in order to avoid uncontrolled congestion of the actual capacity.

Congestion notification communicates a real number bounded by the range [0 , 1]. This ties in with the most well-understood measure of congestion notification: drop probability.

Explicit and Implicit Notification: The byte vs. packet dilemma concerns congestion notification irrespective of whether it is signalled implicitly by drop or explicitly using ECN [RFC3168] or PCN [RFC5670]. Throughout this document, unless clear from the context, the term 'marking' will be used to mean notifying congestion explicitly, while 'congestion notification' will be used to mean notifying congestion either implicitly by drop or explicitly by marking.

Bit-congestible vs. Packet-congestible: If the load on a resource depends on the rate at which packets arrive, it is called 'packet-congestible'. If the load depends on the rate at which bits arrive, it is called 'bit-congestible'.

Examples of packet-congestible resources are route look-up engines and firewalls, because load depends on how many packet headers they have to process. Examples of bit-congestible resources are transmission links, radio power, and most buffer memory, because the load depends on how many bits they have to transmit or store. Some machine architectures use fixed-size packet buffers, so buffer memory in these cases is packet-congestible (see Section 4.1.1).

The path through a machine will typically encounter both packet-congestible and bit-congestible resources. However, currently, a design goal of network processing equipment such as routers and firewalls is to size the packet-processing engine(s) relative to the lines in order to keep packet processing uncongested, even under worst-case packet rates with runs of minimum-size packets. Therefore, packet congestion is currently rare (see Section 3.3 of [RFC6077]), but there is no guarantee that it will not become more common in the future.

Note that information is generally processed or transmitted with a minimum granularity greater than a bit (e.g., octets). The appropriate granularity for the resource in question should be used, but for the sake of brevity we will talk in terms of bytes in this memo.

Coarser Granularity: Resources may be congestible at higher levels of granularity than bits or packets, for instance stateful firewalls are flow-congestible and call-servers are session-congestible. This memo focuses on congestion of connectionless resources, but the same principles may be applicable for congestion notification protocols controlling per-flow and per-session processing or state.

RED Terminology: In RED, whether to use packets or bytes when measuring queues is called, respectively, 'packet-mode queue measurement' or 'byte-mode queue measurement'. And whether the probability of dropping a particular packet is independent or dependent on its size is called, respectively, 'packet-mode drop' or 'byte-mode drop'. The terms 'byte-mode' and 'packet-mode' should not be used without specifying whether they apply to queue measurement or to drop.

1.2. Example Comparing Packet-Mode Drop and Byte-Mode Drop

Taking RED as a well-known example algorithm, a central question addressed by this document is whether to recommend RED's packet-mode drop variant and to deprecate byte-mode drop. Table 1 compares how packet-mode and byte-mode drop affect two flows of different size

packets. For each it gives the expected number of packets and of bits dropped in one second. Each example flow runs at the same bit rate of 48 Mbps, but one is broken up into small 60 byte packets and the other into large 1,500 byte packets.

To keep up the same bit rate, in one second there are about 25 times more small packets because they are 25 times smaller. As can be seen from the table, the packet rate is 100,000 small packets versus 4,000 large packets per second (pps).

Parameter	Formula	Small packets	Large packets
Packet size	s/8	60 B	1,500 B
Packet size	s	480 b	12,000 b
Bit rate	x	48 Mbps	48 Mbps
Packet rate	u = x/s	100 kpps	4 kpps
Packet-mode Drop			
Pkt-loss probability	p	0.1%	0.1%
Pkt-loss rate	p*u	100 pps	4 pps
Bit-loss rate	p*u*s	48 kbps	48 kbps
Byte-mode Drop	MTU, M=12,000 b		
Pkt-loss probability	b = p*s/M	0.004%	0.1%
Pkt-loss rate	b*u	4 pps	4 pps
Bit-loss rate	b*u*s	1.92 kbps	48 kbps

Table 1: Example Comparing Packet-Mode and Byte-Mode Drop

For packet-mode drop, we illustrate the effect of a drop probability of 0.1%, which the algorithm applies to all packets irrespective of size. Because there are 25 times more small packets in one second, it naturally drops 25 times more small packets, that is, 100 small packets but only 4 large packets. But if we count how many bits it drops, there are 48,000 bits in 100 small packets and 48,000 bits in 4 large packets -- the same number of bits of small packets as large.

> The packet-mode drop algorithm drops any bit with the same probability whether the bit is in a small or a large packet.

For byte-mode drop, again we use an example drop probability of 0.1%, but only for maximum size packets (assuming the link maximum transmission unit (MTU) is 1,500 B or 12,000 b). The byte-mode algorithm reduces the drop probability of smaller packets proportional to their size, making the probability that it drops a small packet 25 times smaller at 0.004%. But there are 25 times more small packets, so dropping them with 25 times lower probability results in dropping the same number of packets: 4 drops in both

cases. The 4 small dropped packets contain 25 times less bits than the 4 large dropped packets: 1,920 compared to 48,000.

> The byte-mode drop algorithm drops any bit with a probability proportionate to the size of the packet it is in.

2. Recommendations

This section gives recommendations related to network equipment in Sections 2.1 and 2.2, and we discuss the implications on transport protocols in Sections 2.3 and 2.4.

2.1. Recommendation on Queue Measurement

Ideally, an AQM would measure the service time of the queue to measure congestion of a resource. However service time can only be measured as packets leave the queue, where it is not always expedient to implement a full AQM algorithm. To predict the service time as packets join the queue, an AQM algorithm needs to measure the length of the queue.

In this case, if the resource is bit-congestible, the AQM implementation SHOULD measure the length of the queue in bytes and, if the resource is packet-congestible, the implementation SHOULD measure the length of the queue in packets. Subject to the exceptions below, no other choice makes sense, because the number of packets waiting in the queue isn't relevant if the resource gets congested by bytes and vice versa. For example, the length of the queue into a transmission line would be measured in bytes, while the length of the queue into a firewall would be measured in packets.

To avoid the pathological effects of tail drop, the AQM can then transform this service time or queue length into the probability of dropping or marking a packet (e.g., RED's piecewise linear function between thresholds).

What this advice means for RED as a specific example:

1. A RED implementation SHOULD use byte-mode queue measurement for measuring the congestion of bit-congestible resources and packet-mode queue measurement for packet-congestible resources.

2. An implementation SHOULD NOT make it possible to configure the way a queue measures itself, because whether a queue is bit-congestible or packet-congestible is an inherent property of the queue.

Exceptions to these recommendations might be necessary, for instance where a packet-congestible resource has to be configured as a proxy bottleneck for a bit-congestible resource in an adjacent box that does not support AQM.

The recommended approach in less straightforward scenarios, such as fixed-size packet buffers, resources without a queue, and buffers comprising a mix of packet and bit-congestible resources, is discussed in Section 4.1. For instance, Section 4.1.1 explains that the queue into a line should be measured in bytes even if the queue consists of fixed-size packet buffers, because the root cause of any congestion is bytes arriving too fast for the line -- packets filling buffers are merely a symptom of the underlying congestion of the line.

2.2. Recommendation on Encoding Congestion Notification

When encoding congestion notification (e.g., by drop, ECN, or PCN), the probability that network equipment drops or marks a particular packet to notify congestion SHOULD NOT depend on the size of the packet in question. As the example in Section 1.2 illustrates, to drop any bit with probability 0.1%, it is only necessary to drop every packet with probability 0.1% without regard to the size of each packet.

This approach ensures the network layer offers sufficient congestion information for all known and future transport protocols and also ensures no perverse incentives are created that would encourage transports to use inappropriately small packet sizes.

What this advice means for RED as a specific example:

1. The RED AQM algorithm SHOULD NOT use byte-mode drop, i.e., it ought to use packet-mode drop. Byte-mode drop is more complex, it creates the perverse incentive to fragment segments into tiny pieces and it is vulnerable to floods of small packets.

2. If a vendor has implemented byte-mode drop, and an operator has turned it on, it is RECOMMENDED that the operator use packet-mode drop instead, after establishing if there are any implications on the relative performance of applications using different packet sizes. The unlikely possibility of some application-specific legacy use of byte-mode drop is the only reason that all the above recommendations on encoding congestion notification are not phrased more strongly.

RED as a whole SHOULD NOT be switched off. Without RED, a tail-drop queue biases against large packets and is vulnerable to floods of small packets.

Note well that RED's byte-mode queue drop is completely orthogonal to byte-mode queue measurement and should not be confused with it. If a RED implementation has a byte-mode but does not specify what sort of byte-mode, it is most probably byte-mode queue measurement, which is fine. However, if in doubt, the vendor should be consulted.

A survey (Appendix A) showed that there appears to be little, if any, installed base of the byte-mode drop variant of RED. This suggests that deprecating byte-mode drop will have little, if any, incremental deployment impact.

2.3. Recommendation on Responding to Congestion

When a transport detects that a packet has been lost or congestion marked, it SHOULD consider the strength of the congestion indication as proportionate to the size in octets (bytes) of the missing or marked packet.

In other words, when a packet indicates congestion (by being lost or marked), it can be considered conceptually as if there is a congestion indication on every octet of the packet, not just one indication per packet.

To be clear, the above recommendation solely describes how a transport should interpret the meaning of a congestion indication, as a long term goal. It makes no recommendation on whether a transport should act differently based on this interpretation. It merely aids interoperability between transports, if they choose to make their actions depend on the strength of congestion indications.

This definition will be useful as the IETF transport area continues its programme of:

- updating host-based congestion control protocols to take packet size into account, and
- making transports less sensitive to losing control packets like SYNs and pure ACKs.

What this advice means for the case of TCP:

1. If two TCP flows with different packet sizes are required to run at equal bit rates under the same path conditions, this SHOULD be done by altering TCP (Section 4.2.2), not network equipment (the latter affects other transports besides TCP).

2. If it is desired to improve TCP performance by reducing the chance that a SYN or a pure ACK will be dropped, this SHOULD be done by modifying TCP (Section 4.2.3), not network equipment.

To be clear, we are not recommending at all that TCPs under equivalent conditions should aim for equal bit rates. We are merely saying that anyone trying to do such a thing should modify their TCP algorithm, not the network.

These recommendations are phrased as 'SHOULD' rather than 'MUST', because there may be cases where expediency dictates that compatibility with pre-existing versions of a transport protocol make the recommendations impractical.

2.4. Recommendation on Handling Congestion Indications When Splitting or Merging Packets

Packets carrying congestion indications may be split or merged in some circumstances (e.g., at an RTP / RTP Control Protocol (RTCP) transcoder or during IP fragment reassembly). Splitting and merging only make sense in the context of ECN, not loss.

The general rule to follow is that the number of octets in packets with congestion indications SHOULD be equivalent before and after merging or splitting. This is based on the principle used above; that an indication of congestion on a packet can be considered as an indication of congestion on each octet of the packet.

The above rule is not phrased with the word 'MUST' to allow the following exception. There are cases in which pre-existing protocols were not designed to conserve congestion-marked octets (e.g., IP fragment reassembly [RFC3168] or loss statistics in RTCP receiver reports [RFC3550] before ECN was added [RFC6679]). When any such protocol is updated, it SHOULD comply with the above rule to conserve marked octets. However, the rule may be relaxed if it would otherwise become too complex to interoperate with pre-existing implementations of the protocol.

One can think of a splitting or merging process as if all the incoming congestion-marked octets increment a counter and all the outgoing marked octets decrement the same counter. In order to

ensure that congestion indications remain timely, even the smallest positive remainder in the conceptual counter should trigger the next outgoing packet to be marked (causing the counter to go negative).

3. Motivating Arguments

This section is informative. It justifies the recommendations made in the previous section.

3.1. Avoiding Perverse Incentives to (Ab)use Smaller Packets

Increasingly, it is being recognised that a protocol design must take care not to cause unintended consequences by giving the parties in the protocol exchange perverse incentives [Evol_cc] [RFC3426]. Given there are many good reasons why larger path maximum transmission units (PMTUs) would help solve a number of scaling issues, we do not want to create any bias against large packets that is greater than their true cost.

Imagine a scenario where the same bit rate of packets will contribute the same to bit congestion of a link irrespective of whether it is sent as fewer larger packets or more smaller packets. A protocol design that caused larger packets to be more likely to be dropped than smaller ones would be dangerous in both of the following cases:

Malicious transports: A queue that gives an advantage to small packets can be used to amplify the force of a flooding attack. By sending a flood of small packets, the attacker can get the queue to discard more large-packet traffic, allowing more attack traffic to get through to cause further damage. Such a queue allows attack traffic to have a disproportionately large effect on regular traffic without the attacker having to do much work.

Non-malicious transports: Even if an application designer is not actually malicious, if over time it is noticed that small packets tend to go faster, designers will act in their own interest and use smaller packets. Queues that give advantage to small packets create an evolutionary pressure for applications or transports to send at the same bit rate but break their data stream down into tiny segments to reduce their drop rate. Encouraging a high volume of tiny packets might in turn unnecessarily overload a completely unrelated part of the system, perhaps more limited by header processing than bandwidth.

Imagine that two unresponsive flows arrive at a bit-congestible transmission link each with the same bit rate, say 1 Mbps, but one consists of 1,500 B and the other 60 B packets, which are 25x smaller. Consider a scenario where gentle RED [gentle_RED] is used,

along with the variant of RED we advise against, i.e., where the RED algorithm is configured to adjust the drop probability of packets in proportion to each packet's size (byte-mode packet drop). In this case, RED aims to drop 25x more of the larger packets than the smaller ones. Thus, for example, if RED drops 25% of the larger packets, it will aim to drop 1% of the smaller packets (but, in practice, it may drop more as congestion increases; see Appendix B.4 of [RFC4828]). Even though both flows arrive with the same bit rate, the bit rate the RED queue aims to pass to the line will be 750 kbps for the flow of larger packets but 990 kbps for the smaller packets (because of rate variations, it will actually be a little less than this target).

Note that, although the byte-mode drop variant of RED amplifies small-packet attacks, tail-drop queues amplify small-packet attacks even more (see Security Considerations in Section 6). Wherever possible, neither should be used.

3.2. Small != Control

Dropping fewer control packets considerably improves performance. It is tempting to drop small packets with lower probability in order to improve performance, because many control packets tend to be smaller (TCP SYNs and ACKs, DNS queries and responses, SIP messages, HTTP GETs, etc). However, we must not give control packets preference purely by virtue of their smallness, otherwise it is too easy for any data source to get the same preferential treatment simply by sending data in smaller packets. Again, we should not create perverse incentives to favour small packets rather than to favour control packets, which is what we intend.

Just because many control packets are small does not mean all small packets are control packets.

So, rather than fix these problems in the network, we argue that the transport should be made more robust against losses of control packets (see Section 4.2.3).

3.3. Transport-Independent Network

TCP congestion control ensures that flows competing for the same resource each maintain the same number of segments in flight, irrespective of segment size. So under similar conditions, flows with different segment sizes will get different bit rates.

To counter this effect, it seems tempting not to follow our recommendation, and instead for the network to bias congestion notification by packet size in order to equalise the bit rates of

flows with different packet sizes. However, in order to do this, the queuing algorithm has to make assumptions about the transport, which become embedded in the network. Specifically:

- The queuing algorithm has to assume how aggressively the transport will respond to congestion (see Section 4.2.4). If the network assumes the transport responds as aggressively as TCP NewReno, it will be wrong for Compound TCP and differently wrong for Cubic TCP, etc. To achieve equal bit rates, each transport then has to guess what assumption the network made, and work out how to replace this assumed aggressiveness with its own aggressiveness.

- Also, if the network biases congestion notification by packet size, it has to assume a baseline packet size -- all proposed algorithms use the local MTU (for example, see the byte-mode loss probability formula in Table 1). Then if the non-Reno transports mentioned above are trying to reverse engineer what the network assumed, they also have to guess the MTU of the congested link.

Even though reducing the drop probability of small packets (e.g., RED's byte-mode drop) helps ensure TCP flows with different packet sizes will achieve similar bit rates, we argue that this correction should be made to any future transport protocols based on TCP, not to the network in order to fix one transport, no matter how predominant it is. Effectively, favouring small packets is reverse engineering of network equipment around one particular transport protocol (TCP), contrary to the excellent advice in [RFC3426], which asks designers to question "Why are you proposing a solution at this layer of the protocol stack, rather than at another layer?"

In contrast, if the network never takes packet size into account, the transport can be certain it will never need to guess any assumptions that the network has made. And the network passes two pieces of information to the transport that are sufficient in all cases: i) congestion notification on the packet and ii) the size of the packet. Both are available for the transport to combine (by taking packet size into account when responding to congestion) or not. Appendix B checks that these two pieces of information are sufficient for all relevant scenarios.

When the network does not take packet size into account, it allows transport protocols to choose whether or not to take packet size into account. However, if the network were to bias congestion notification by packet size, transport protocols would have no choice; those that did not take into account packet size themselves would unwittingly become dependent on packet size, and those that already took packet size into account would end up taking it into account twice.

3.4. Partial Deployment of AQM

In overview, the argument in this section runs as follows:

- Because the network does not and cannot always drop packets in proportion to their size, it shouldn't be given the task of making drop signals depend on packet size at all.

- Transports on the other hand don't always want to make their rate response proportional to the size of dropped packets, but if they want to, they always can.

The argument is similar to the end-to-end argument that says "Don't do X in the network if end systems can do X by themselves, and they want to be able to choose whether to do X anyway". Actually the following argument is stronger; in addition it says "Don't give the network task X that could be done by the end systems, if X is not deployed on all network nodes, and end systems won't be able to tell whether their network is doing X, or whether they need to do X themselves." In this case, the X in question is "making the response to congestion depend on packet size".

We will now re-run this argument reviewing each step in more depth. The argument applies solely to drop, not to ECN marking.

A queue drops packets for either of two reasons: a) to signal to host congestion controls that they should reduce the load and b) because there is no buffer left to store the packets. Active queue management tries to use drops as a signal for hosts to slow down (case a) so that drops due to buffer exhaustion (case b) should not be necessary.

AQM is not universally deployed in every queue in the Internet; many cheap Ethernet bridges, software firewalls, NATs on consumer devices, etc implement simple tail-drop buffers. Even if AQM were universal, it has to be able to cope with buffer exhaustion (by switching to a behaviour like tail drop), in order to cope with unresponsive or excessive transports. For these reasons networks will sometimes be dropping packets as a last resort (case b) rather than under AQM control (case a).

When buffers are exhausted (case b), they don't naturally drop packets in proportion to their size. The network can only reduce the probability of dropping smaller packets if it has enough space to store them somewhere while it waits for a larger packet that it can drop. If the buffer is exhausted, it does not have this choice. Admittedly tail drop does naturally drop somewhat fewer small packets, but exactly how few depends more on the mix of sizes than

the size of the packet in question. Nonetheless, in general, if we wanted networks to do size-dependent drop, we would need universal deployment of (packet-size dependent) AQM code, which is currently unrealistic.

A host transport cannot know whether any particular drop was a deliberate signal from an AQM or a sign of a queue shedding packets due to buffer exhaustion. Therefore, because the network cannot universally do size-dependent drop, it should not do it all.

Whereas universality is desirable in the network, diversity is desirable between different transport-layer protocols -- some, like standards track TCP congestion control [RFC5681], may not choose to make their rate response proportionate to the size of each dropped packet, while others will (e.g., TCP-Friendly Rate Control for Small Packets (TFRC-SP) [RFC4828]).

3.5. Implementation Efficiency

Biasing against large packets typically requires an extra multiply and divide in the network (see the example byte-mode drop formula in Table 1). Taking packet size into account at the transport rather than in the network ensures that neither the network nor the transport needs to do a multiply operation -- multiplication by packet size is effectively achieved as a repeated add when the transport adds to its count of marked bytes as each congestion event is fed to it. Also, the work to do the biasing is spread over many hosts, rather than concentrated in just the congested network element. These aren't principled reasons in themselves, but they are a happy consequence of the other principled reasons.

4. A Survey and Critique of Past Advice

This section is informative, not normative.

The original 1993 paper on RED [RED93] proposed two options for the RED active queue management algorithm: packet mode and byte mode. Packet mode measured the queue length in packets and dropped (or marked) individual packets with a probability independent of their size. Byte mode measured the queue length in bytes and marked an individual packet with probability in proportion to its size (relative to the maximum packet size). In the paper's outline of further work, it was stated that no recommendation had been made on whether the queue size should be measured in bytes or packets, but noted that the difference could be significant.

When RED was recommended for general deployment in 1998 [RFC2309], the two modes were mentioned implying the choice between them was a question of performance, referring to a 1997 email [pktByteEmail] for advice on tuning. A later addendum to this email introduced the insight that there are in fact two orthogonal choices:

- whether to measure queue length in bytes or packets (Section 4.1), and
- whether the drop probability of an individual packet should depend on its own size (Section 4.2).

The rest of this section is structured accordingly.

4.1. Congestion Measurement Advice

The choice of which metric to use to measure queue length was left open in RFC 2309. It is now well understood that queues for bit-congestible resources should be measured in bytes, and queues for packet-congestible resources should be measured in packets [pktByteEmail].

Congestion in some legacy bit-congestible buffers is only measured in packets not bytes. In such cases, the operator has to take into account a typical mix of packet sizes when setting the thresholds. Any AQM algorithm on such a buffer will be oversensitive to high proportions of small packets, e.g., a DoS attack, and under-sensitive to high proportions of large packets. However, there is no need to make allowances for the possibility of such a legacy in future protocol design. This is safe because any under-sensitivity during unusual traffic mixes cannot lead to congestion collapse given that the buffer will eventually revert to tail drop, which discards proportionately more large packets.

4.1.1. Fixed-Size Packet Buffers

The question of whether to measure queues in bytes or packets seems to be well understood. However, measuring congestion is confusing when the resource is bit-congestible but the queue into the resource is packet-congestible. This section outlines the approach to take.

Some, mostly older, queuing hardware allocates fixed-size buffers in which to store each packet in the queue. This hardware forwards packets to the line in one of two ways:

- With some hardware, any fixed-size buffers not completely filled by a packet are padded when transmitted to the wire. This case should clearly be treated as packet-congestible, because both

queuing and transmission are in fixed MTU-size units. Therefore, the queue length in packets is a good model of congestion of the link.

- More commonly, hardware with fixed-size packet buffers transmits packets to the line without padding. This implies a hybrid forwarding system with transmission congestion dependent on the size of packets but queue congestion dependent on the number of packets, irrespective of their size.

  Nonetheless, there would be no queue at all unless the line had become congested -- the root cause of any congestion is too many bytes arriving for the line. Therefore, the AQM should measure the queue length as the sum of all the packet sizes in bytes that are queued up waiting to be serviced by the line, irrespective of whether each packet is held in a fixed-size buffer.

In the (unlikely) first case where use of padding means the queue should be measured in packets, further confusion is likely because the fixed buffers are rarely all one size. Typically, pools of different-sized buffers are provided (Cisco uses the term 'buffer carving' for the process of dividing up memory into these pools [IOSArch]). Usually, if the pool of small buffers is exhausted, arriving small packets can borrow space in the pool of large buffers, but not vice versa. However, there is no need to consider all this complexity, because the root cause of any congestion is still line overload -- buffer consumption is only the symptom. Therefore, the length of the queue should be measured as the sum of the bytes in the queue that will be transmitted to the line, including any padding. In the (unusual) case of transmission with padding, this means the sum of the sizes of the small buffers queued plus the sum of the sizes of the large buffers queued.

We will return to borrowing of fixed-size buffers when we discuss biasing the drop/marking probability of a specific packet because of its size in Section 4.2.1. But here, we can repeat the simple rule for how to measure the length of queues of fixed buffers: no matter how complicated the buffering scheme is, ultimately a transmission line is nearly always bit-congestible so the number of bytes queued up waiting for the line measures how congested the line is, and it is rarely important to measure how congested the buffering system is.

4.1.2. Congestion Measurement without a Queue

AQM algorithms are nearly always described assuming there is a queue for a congested resource and the algorithm can use the queue length to determine the probability that it will drop or mark each packet. But not all congested resources lead to queues. For instance, power-

limited resources are usually bit-congestible if energy is primarily required for transmission rather than header processing, but it is rare for a link protocol to build a queue as it approaches maximum power.

Nonetheless, AQM algorithms do not require a queue in order to work. For instance, spectrum congestion can be modelled by signal quality using the target bit-energy-to-noise-density ratio. And, to model radio power exhaustion, transmission-power levels can be measured and compared to the maximum power available. [ECNFixedWireless] proposes a practical and theoretically sound way to combine congestion notification for different bit-congestible resources at different layers along an end-to-end path, whether wireless or wired, and whether with or without queues.

In wireless protocols that use request to send / clear to send (RTS / CTS) control, such as some variants of IEEE802.11, it is reasonable to base an AQM on the time spent waiting for transmission opportunities (TXOPs) even though the wireless spectrum is usually regarded as congested by bits (for a given coding scheme). This is because requests for TXOPs queue up as the spectrum gets congested by all the bits being transferred. So the time that TXOPs are queued directly reflects bit congestion of the spectrum.

4.2. Congestion Notification Advice

4.2.1. Network Bias When Encoding

4.2.1.1. Advice on Packet-Size Bias in RED

The previously mentioned email [pktByteEmail] referred to by [RFC2309] advised that most scarce resources in the Internet were bit-congestible, which is still believed to be true (Section 1.1). But it went on to offer advice that is updated by this memo. It said that drop probability should depend on the size of the packet being considered for drop if the resource is bit-congestible, but not if it is packet-congestible. The argument continued that if packet drops were inflated by packet size (byte-mode dropping), "a flow’s fraction of the packet drops is then a good indication of that flow’s fraction of the link bandwidth in bits per second". This was consistent with a referenced policing mechanism being worked on at the time for detecting unusually high bandwidth flows, eventually published in 1999 [pBox]. However, the problem could and should have been solved by making the policing mechanism count the volume of bytes randomly dropped, not the number of packets.

A few months before RFC 2309 was published, an addendum was added to the above archived email referenced from the RFC, in which the final paragraph seemed to partially retract what had previously been said. It clarified that the question of whether the probability of dropping/marking a packet should depend on its size was not related to whether the resource itself was bit-congestible, but a completely orthogonal question. However, the only example given had the queue measured in packets but packet drop depended on the size of the packet in question. No example was given the other way round.

In 2000, Cnodder et al. [REDbyte] pointed out that there was an error in the part of the original 1993 RED algorithm that aimed to distribute drops uniformly, because it didn't correctly take into account the adjustment for packet size. They recommended an algorithm called RED_4 to fix this. But they also recommended a further change, RED_5, to adjust the drop rate dependent on the square of the relative packet size. This was indeed consistent with one implied motivation behind RED's byte-mode drop -- that we should reverse engineer the network to improve the performance of dominant end-to-end congestion control mechanisms. This memo makes a different recommendations in Section 2.

By 2003, a further change had been made to the adjustment for packet size, this time in the RED algorithm of the ns2 simulator. Instead of taking each packet's size relative to a 'maximum packet size', it was taken relative to a 'mean packet size', intended to be a static value representative of the 'typical' packet size on the link. We have not been able to find a justification in the literature for this change; however, Eddy and Allman conducted experiments [REDbias] that assessed how sensitive RED was to this parameter, amongst other things. This changed algorithm can often lead to drop probabilities of greater than 1 (which gives a hint that there is probably a mistake in the theory somewhere).

On 10-Nov-2004, this variant of byte-mode packet drop was made the default in the ns2 simulator. It seems unlikely that byte-mode drop has ever been implemented in production networks (Appendix A); therefore, any conclusions based on ns2 simulations that use RED without disabling byte-mode drop are likely to behave very differently from RED in production networks.

4.2.1.2. Packet-Size Bias Regardless of AQM

The byte-mode drop variant of RED (or a similar variant of other AQM algorithms) is not the only possible bias towards small packets in queuing systems. We have already mentioned that tail-drop queues naturally tend to lock out large packets once they are full.

But also, queues with fixed-size buffers reduce the probability that small packets will be dropped if (and only if) they allow small packets to borrow buffers from the pools for larger packets (see Section 4.1.1). Borrowing effectively makes the maximum queue size for small packets greater than that for large packets, because more buffers can be used by small packets while less will fit large packets. Incidentally, the bias towards small packets from buffer borrowing is nothing like as large as that of RED's byte-mode drop.

Nonetheless, fixed-buffer memory with tail drop is still prone to lock out large packets, purely because of the tail-drop aspect. So, fixed-size packet buffers should be augmented with a good AQM algorithm and packet-mode drop. If an AQM is too complicated to implement with multiple fixed buffer pools, the minimum necessary to prevent large-packet lockout is to ensure that smaller packets never use the last available buffer in any of the pools for larger packets.

4.2.2. Transport Bias When Decoding

The above proposals to alter the network equipment to bias towards smaller packets have largely carried on outside the IETF process. Whereas, within the IETF, there are many different proposals to alter transport protocols to achieve the same goals, i.e., either to make the flow bit rate take into account packet size, or to protect control packets from loss. This memo argues that altering transport protocols is the more principled approach.

A recently approved experimental RFC adapts its transport-layer protocol to take into account packet sizes relative to typical TCP packet sizes. This proposes a new small-packet variant of TCP-friendly rate control (TFRC [RFC5348]), which is called TFRC-SP [RFC4828]. Essentially, it proposes a rate equation that inflates the flow rate by the ratio of a typical TCP segment size (1,500 B including TCP header) over the actual segment size [PktSizeEquCC]. (There are also other important differences of detail relative to TFRC, such as using virtual packets [CCvarPktSize] to avoid responding to multiple losses per round trip and using a minimum inter-packet interval.)

Section 4.5.1 of the TFRC-SP specification discusses the implications of operating in an environment where queues have been configured to drop smaller packets with proportionately lower probability than larger ones. But it only discusses TCP operating in such an environment, only mentioning TFRC-SP briefly when discussing how to define fairness with TCP. And it only discusses the byte-mode dropping version of RED as it was before Cnodder et al. pointed out that it didn't sufficiently bias towards small packets to make TCP independent of packet size.

So the TFRC-SP specification doesn't address the issue of whether the network or the transport _should_ handle fairness between different packet sizes. In Appendix B.4 of RFC 4828, it discusses the possibility of both TFRC-SP and some network buffers duplicating each other's attempts to deliberately bias towards small packets. But the discussion is not conclusive, instead reporting simulations of many of the possibilities in order to assess performance but not recommending any particular course of action.

The paper originally proposing TFRC with virtual packets (VP-TFRC) [CCvarPktSize] proposed that there should perhaps be two variants to cater for the different variants of RED. However, as the TFRC-SP authors point out, there is no way for a transport to know whether some queues on its path have deployed RED with byte-mode packet drop (except if an exhaustive survey found that no one has deployed it! -- see Appendix A). Incidentally, VP-TFRC also proposed that byte-mode RED dropping should really square the packet-size compensation factor (like that of Cnodder's RED_5, but apparently unaware of it).

Pre-congestion notification [RFC5670] is an IETF technology to use a virtual queue for AQM marking for packets within one Diffserv class in order to give early warning prior to any real queuing. The PCN-marking algorithms have been designed not to take into account packet size when forwarding through queues. Instead, the general principle has been to take the sizes of marked packets into account when monitoring the fraction of marking at the edge of the network, as recommended here.

4.2.3. Making Transports Robust against Control Packet Losses

Recently, two RFCs have defined changes to TCP that make it more robust against losing small control packets [RFC5562] [RFC5690]. In both cases, they note that the case for these two TCP changes would be weaker if RED were biased against dropping small packets. We argue here that these two proposals are a safer and more principled way to achieve TCP performance improvements than reverse engineering RED to benefit TCP.

Although there are no known proposals, it would also be possible and perfectly valid to make control packets robust against drop by requesting a scheduling class with lower drop probability, which would be achieved by re-marking to a Diffserv code point [RFC2474] within the same behaviour aggregate.

Although not brought to the IETF, a simple proposal from Wischik [DupTCP] suggests that the first three packets of every TCP flow should be routinely duplicated after a short delay. It shows that this would greatly improve the chances of short flows completing

quickly, but it would hardly increase traffic levels on the Internet, because Internet bytes have always been concentrated in the large flows. It further shows that the performance of many typical applications depends on completion of long serial chains of short messages. It argues that, given most of the value people get from the Internet is concentrated within short flows, this simple expedient would greatly increase the value of the best-effort Internet at minimal cost. A similar but more extensive approach has been evaluated on Google servers [GentleAggro].

The proposals discussed in this sub-section are experimental approaches that are not yet in wide operational use, but they are existence proofs that transports can make themselves robust against loss of control packets. The examples are all TCP-based, but applications over non-TCP transports could mitigate loss of control packets by making similar use of Diffserv, data duplication, FEC, etc.

4.2.4. Congestion Notification: Summary of Conflicting Advice

transport cc	RED_1 (packet-mode drop)	RED_4 (linear byte-mode drop)	RED_5 (square byte-mode drop)
TCP or TFRC	s/sqrt(p)	sqrt(s/p)	1/sqrt(p)
TFRC-SP	1/sqrt(p)	1/sqrt(s*p)	1/(s*sqrt(p))

Table 2: Dependence of flow bit rate per RTT on packet size, s, and drop probability, p, when there is network and/or transport bias towards small packets to varying degrees

Table 2 aims to summarise the potential effects of all the advice from different sources. Each column shows a different possible AQM behaviour in different queues in the network, using the terminology of Cnodder et al. outlined earlier (RED_1 is basic RED with packet-mode drop). Each row shows a different transport behaviour: TCP [RFC5681] and TFRC [RFC5348] on the top row with TFRC-SP [RFC4828] below. Each cell shows how the bits per round trip of a flow depends on packet size, s, and drop probability, p. In order to declutter the formulae to focus on packet-size dependence, they are all given per round trip, which removes any RTT term.

Let us assume that the goal is for the bit rate of a flow to be independent of packet size. Suppressing all inessential details, the table shows that this should either be achievable by not altering the TCP transport in a RED_5 network, or using the small packet TFRC-SP

transport (or similar) in a network without any byte-mode dropping RED (top right and bottom left). Top left is the 'do nothing' scenario, while bottom right is the 'do both' scenario in which the bit rate would become far too biased towards small packets. Of course, if any form of byte-mode dropping RED has been deployed on a subset of queues that congest, each path through the network will present a different hybrid scenario to its transport.

Whatever the case, we can see that the linear byte-mode drop column in the middle would considerably complicate the Internet. Even if one believes the network should be doing the biasing, linear byte-mode drop is a half-way house that doesn't bias enough towards small packets. Section 2 recommends that _all_ bias in network equipment towards small packets should be turned off -- if indeed any equipment vendors have implemented it -- leaving packet-size bias solely as the preserve of the transport layer (solely the leftmost, packet-mode drop column).

In practice, it seems that no deliberate bias towards small packets has been implemented for production networks. Of the 19% of vendors who responded to a survey of 84 equipment vendors, none had implemented byte-mode drop in RED (see Appendix A for details).

5. Outstanding Issues and Next Steps

5.1. Bit-congestible Network

For a connectionless network with nearly all resources being bit-congestible, the recommended position is clear -- the network should not make allowance for packet sizes and the transport should. This leaves two outstanding issues:

- The question of how to handle any legacy AQM deployments using byte-mode drop;
- The need to start a programme to update transport congestion control protocol standards to take packet size into account.

A survey of equipment vendors (Section 4.2.4) found no evidence that byte-mode packet drop had been implemented, so deployment will be sparse at best. A migration strategy is not really needed to remove an algorithm that may not even be deployed.

A programme of experimental updates to take packet size into account in transport congestion control protocols has already started with TFRC-SP [RFC4828].

5.2. Bit- and Packet-Congestible Network

The position is much less clear-cut if the Internet becomes populated by a more even mix of both packet-congestible and bit-congestible resources (see Appendix B.2). This problem is not pressing, because most Internet resources are designed to be bit-congestible before packet processing starts to congest (see Section 1.1).

The IRTF's Internet Congestion Control Research Group (ICCRG) has set itself the task of reaching consensus on generic forwarding mechanisms that are necessary and sufficient to support the Internet's future congestion control requirements (the first challenge in [RFC6077]). The research question of whether packet congestion might become common and what to do if it does may in the future be explored in the IRTF (the "Challenge 3: Packet Size" in [RFC6077]).

Note that sometimes it seems that resources might be congested by neither bits nor packets, e.g., where the queue for access to a wireless medium is in units of transmission opportunities. However, the root cause of congestion of the underlying spectrum is overload of bits (see Section 4.1.2).

6. Security Considerations

This memo recommends that queues do not bias drop probability due to packets size. For instance, dropping small packets less often than large ones creates a perverse incentive for transports to break down their flows into tiny segments. One of the benefits of implementing AQM was meant to be to remove this perverse incentive that tail-drop queues gave to small packets.

In practice, transports cannot all be trusted to respond to congestion. So another reason for recommending that queues not bias drop probability towards small packets is to avoid the vulnerability to small-packet DDoS attacks that would otherwise result. One of the benefits of implementing AQM was meant to be to remove tail drop's DoS vulnerability to small packets, so we shouldn't add it back again.

If most queues implemented AQM with byte-mode drop, the resulting network would amplify the potency of a small-packet DDoS attack. At the first queue, the stream of packets would push aside a greater proportion of large packets, so more of the small packets would survive to attack the next queue. Thus a flood of small packets would continue on towards the destination, pushing regular traffic with large packets out of the way in one queue after the next, but suffering much less drop itself.

Appendix C explains why the ability of networks to police the response of _any_ transport to congestion depends on bit-congestible network resources only doing packet-mode drop, not byte-mode drop. In summary, it says that making drop probability depend on the size of the packets that bits happen to be divided into simply encourages the bits to be divided into smaller packets. Byte-mode drop would therefore irreversibly complicate any attempt to fix the Internet's incentive structures.

7. Conclusions

This memo identifies the three distinct stages of the congestion notification process where implementations need to decide whether to take packet size into account. The recommendations provided in Section 2 of this memo are different in each case:

- When network equipment measures the length of a queue, if it is not feasible to use time; it is recommended to count in bytes if the network resource is congested by bytes, or to count in packets if is congested by packets.

- When network equipment decides whether to drop (or mark) a packet, it is recommended that the size of the particular packet should not be taken into account.

- However, when a transport algorithm responds to a dropped or marked packet, the size of the rate reduction should be proportionate to the size of the packet.

In summary, the answers are 'it depends', 'no', and 'yes', respectively.

For the specific case of RED, this means that byte-mode queue measurement will often be appropriate, but the use of byte-mode drop is very strongly discouraged.

At the transport layer, the IETF should continue updating congestion control protocols to take into account the size of each packet that indicates congestion. Also, the IETF should continue to make protocols less sensitive to losing control packets like SYNs, pure ACKs, and DNS exchanges. Although many control packets happen to be small, the alternative of network equipment favouring all small packets would be dangerous. That would create perverse incentives to split data transfers into smaller packets.

The memo develops these recommendations from principled arguments concerning scaling, layering, incentives, inherent efficiency, security, and 'policeability'. It also addresses practical issues

such as specific buffer architectures and incremental deployment. Indeed, a limited survey of RED implementations is discussed, which shows there appears to be little, if any, installed base of RED's byte-mode drop. Therefore, it can be deprecated with little, if any, incremental deployment complications.

The recommendations have been developed on the well-founded basis that most Internet resources are bit-congestible, not packet-congestible. We need to know the likelihood that this assumption will prevail in the longer term and, if it might not, what protocol changes will be needed to cater for a mix of the two. The IRTF Internet Congestion Control Research Group (ICCRG) is currently working on these problems [RFC6077].

8. Acknowledgements

Thank you to Sally Floyd, who gave extensive and useful review comments. Also thanks for the reviews from Philip Eardley, David Black, Fred Baker, David Taht, Toby Moncaster, Arnaud Jacquet, and Mirja Kuehlewind, as well as helpful explanations of different hardware approaches from Larry Dunn and Fred Baker. We are grateful to Bruce Davie and his colleagues for providing a timely and efficient survey of RED implementation in Cisco's product range. Also, grateful thanks to Toby Moncaster, Will Dormann, John Regnault, Simon Carter, and Stefaan De Cnodder who further helped survey the current status of RED implementation and deployment, and, finally, thanks to the anonymous individuals who responded.

Bob Briscoe and Jukka Manner were partly funded by Trilogy and Trilogy 2, research projects (ICT-216372, ICT-317756) supported by the European Community under its Seventh Framework Programme. The views expressed here are those of the authors only.

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.

[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.

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, September 2000.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

9.2. Informative References

[BLUE02] Feng, W-c., Shin, K., Kandlur, D., and D. Saha, "The BLUE active queue management algorithms", IEEE/ACM Transactions on Networking 10(4) 513-528, August 2002, <http://dx.doi.org/10.1109/TNET.2002.801399>.

[CCvarPktSize] Widmer, J., Boutremans, C., and J-Y. Le Boudec, "End-to-end congestion control for TCP-friendly flows with variable packet size", ACM CCR 34(2) 137-151, April 2004, <http://doi.acm.org/10.1145/997150.997162>.

[CHOKe_Var_Pkt] Psounis, K., Pan, R., and B. Prabhaker, "Approximate Fair Dropping for Variable-Length Packets", IEEE Micro 21(1):48-56, January-February 2001, <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=903061>.

[CoDel] Nichols, K. and V. Jacobson, "Controlled Delay Active Queue Management", Work in Progress, February 2013.

[DRQ] Shin, M., Chong, S., and I. Rhee, "Dual-Resource TCP/AQM for Processing-Constrained Networks", IEEE/ACM Transactions on Networking Vol 16, issue 2, April 2008, <http://dx.doi.org/10.1109/TNET.2007.900415>.

[DupTCP] Wischik, D., "Short messages", Philosophical Transactions of the Royal Society A 366(1872):1941-1953, June 2008, <http://rsta.royalsocietypublishing.org/content/366/1872/1941.full.pdf+html>.

[ECNFixedWireless] Siris, V., "Resource Control for Elastic Traffic in CDMA Networks", Proc. ACM MOBICOM'02 , September 2002, <http://www.ics.forth.gr/netlab/publications/resource_control_elastic_cdma.html>.

[Evol_cc] Gibbens, R. and F. Kelly, "Resource pricing and the evolution of congestion control", Automatica 35(12)1969-1985, December 1999, <http://www.sciencedirect.com/science/article/pii/S0005109899001351>.

[GentleAggro] Flach, T., Dukkipati, N., Terzis, A., Raghavan, B., Cardwell, N., Cheng, Y., Jain, A., Hao, S., Katz-Bassett, E., and R. Govindan, "Reducing web latency: the virtue of gentle aggression", ACM SIGCOMM CCR 43(4)159-170, August 2013, <http://doi.acm.org/10.1145/2486001.2486014>.

[IOSArch] Bollapragada, V., White, R., and C. Murphy, "Inside Cisco IOS Software Architecture", Cisco Press: CCIE Professional Development ISBN13: 978-1-57870-181-0, July 2000.

[PIE] Pan, R., Natarajan, P., Piglione, C., Prabhu, M., Subramanian, V., Baker, F., and B. Steeg, "PIE: A Lightweight Control Scheme To Address the Bufferbloat Problem", Work in Progress, February 2014.

[PktSizeEquCC] Vasallo, P., "Variable Packet Size Equation-Based Congestion Control", ICSI Technical Report tr-00-008, 2000, <http://http.icsi.berkeley.edu/ftp/global/pub/techreports/2000/tr-00-008.pdf>.

[RED93] Floyd, S. and V. Jacobson, "Random Early Detection (RED) gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking 1(4) 397--413, August 1993, <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=251892>.

[REDbias] Eddy, W. and M. Allman, "A Comparison of RED's Byte and Packet Modes", Computer Networks 42(3) 261--280, June 2003, <http://www.ir.bbn.com/documents/articles/redbias.ps>.

[REDbyte] De Cnodder, S., Elloumi, O., and K. Pauwels, "Effect of different packet sizes on RED performance", Proc. 5th IEEE Symposium on Computers and Communications (ISCC) 793-799, July 2000, <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=860741>.

[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.

[RFC3426] Floyd, S., "General Architectural and Policy Considerations", RFC 3426, November 2002.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC3714] Floyd, S. and J. Kempf, "IAB Concerns Regarding Congestion Control for Voice Traffic in the Internet", RFC 3714, March 2004.

[RFC4828] Floyd, S. and E. Kohler, "TCP Friendly Rate Control (TFRC): The Small-Packet (SP) Variant", RFC 4828, April 2007.

[RFC5348] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 5348, September 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.

[RFC5670] Eardley, P., "Metering and Marking Behaviour of PCN-Nodes", RFC 5670, November 2009.

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

[RFC5690] Floyd, S., Arcia, A., Ros, D., and J. Iyengar, "Adding Acknowledgement Congestion Control to TCP", RFC 5690, February 2010.

[RFC6077] Papadimitriou, D., Welzl, M., Scharf, M., and B. Briscoe, "Open Research Issues in Internet Congestion Control", RFC 6077, February 2011.

[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.

[Rate_fair_Dis] Briscoe, B., "Flow Rate Fairness: Dismantling a Religion", ACM CCR 37(2)63-74, April 2007, <http://portal.acm.org/citation.cfm?id=1232926>.

[gentle_RED] Floyd, S., "Recommendation on using the "gentle_" variant of RED", Web page , March 2000, <http://www.icir.org/floyd/red/gentle.html>.

[pBox] Floyd, S. and K. Fall, "Promoting the Use of End-to-End Congestion Control", IEEE/ACM Transactions on Networking 7(4) 458--472, August 1999, <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=793002>.

[pktByteEmail] Floyd, S., "RED: Discussions of Byte and Packet Modes", email, March 1997, <http://ee.lbl.gov/floyd/REDaveraging.txt>.

Appendix A. Survey of RED Implementation Status

This Appendix is informative, not normative.

In May 2007 a survey was conducted of 84 vendors to assess how widely drop probability based on packet size has been implemented in RED Table 3. About 19% of those surveyed replied, giving a sample size of 16. Although in most cases we do not have permission to identify the respondents, we can say that those that have responded include most of the larger equipment vendors, covering a large fraction of the market. The two who gave permission to be identified were Cisco and Alcatel-Lucent. The others range across the large network equipment vendors at L3 & L2, firewall vendors, wireless equipment vendors, as well as large software businesses with a small selection of networking products. All those who responded confirmed that they have not implemented the variant of RED with drop dependent on packet size (2 were fairly sure they had not but needed to check more thoroughly). At the time the survey was conducted, Linux did not implement RED with packet-size bias of drop, although we have not investigated a wider range of open source code.

Response	No. of vendors	% of vendors
Not implemented	14	17%
Not implemented (probably)	2	2%
Implemented	0	0%
No response	68	81%
Total companies/orgs surveyed	84	100%

Table 3: Vendor Survey on byte-mode drop variant of RED (lower drop probability for small packets)

Where reasons were given for why the byte-mode drop variant had not been implemented, the extra complexity of packet-bias code was most prevalent, though one vendor had a more principled reason for avoiding it -- similar to the argument of this document.

Our survey was of vendor implementations, so we cannot be certain about operator deployment. But we believe many queues in the Internet are still tail drop. The company of one of the co-authors (BT) has widely deployed RED; however, many tail-drop queues are bound to still exist, particularly in access network equipment and on middleboxes like firewalls, where RED is not always available.

Routers using a memory architecture based on fixed-size buffers with borrowing may also still be prevalent in the Internet. As explained in Section 4.2.1, these also provide a marginal (but legitimate) bias towards small packets. So even though RED byte-mode drop is not prevalent, it is likely there is still some bias towards small packets in the Internet due to tail-drop and fixed-buffer borrowing.

Appendix B. Sufficiency of Packet-Mode Drop

This Appendix is informative, not normative.

Here we check that packet-mode drop (or marking) in the network gives sufficiently generic information for the transport layer to use. We check against a 2x2 matrix of four scenarios that may occur now or in the future (Table 4). Checking the two scenarios in each of the horizontal and vertical dimensions tests the extremes of sensitivity to packet size in the transport and in the network respectively.

Note that this section does not consider byte-mode drop at all. Having deprecated byte-mode drop, the goal here is to check that packet-mode drop will be sufficient in all cases.

Transport -> / Network	a) Independent of packet size of congestion notifications	b) Dependent on packet size of congestion notifications
1) Predominantly bit-congestible network	Scenario a1)	Scenario b1)
2) Mix of bit-congestible and pkt-congestible network	Scenario a2)	Scenario b2)

Table 4: Four Possible Congestion Scenarios

Appendix B.1 focuses on the horizontal dimension of Table 4 checking that packet-mode drop (or marking) gives sufficient information, whether or not the transport uses it -- scenarios b) and a) respectively.

Appendix B.2 focuses on the vertical dimension of Table 4, checking that packet-mode drop gives sufficient information to the transport whether resources in the network are bit-congestible or packet-congestible (these terms are defined in Section 1.1).

Notation: To be concrete, we will compare two flows with different packet sizes, s_1 and s_2. As an example, we will take s_1 = 60 B = 480 b and s_2 = 1,500 B = 12,000 b.

A flow's bit rate, x [bps], is related to its packet rate, u [pps], by

$$x(t) = s*u(t).$$

In the bit-congestible case, path congestion will be denoted by p_b, and in the packet-congestible case by p_p. When either case is implied, the letter p alone will denote path congestion.

B.1. Packet-Size (In)Dependence in Transports

In all cases, we consider a packet-mode drop queue that indicates congestion by dropping (or marking) packets with probability p irrespective of packet size. We use an example value of loss (marking) probability, p=0.1%.

A transport like TCP as specified in RFC 5681 treats a congestion notification on any packet whatever its size as one event. However, a network with just the packet-mode drop algorithm gives more information if the transport chooses to use it. We will use Table 5 to illustrate this.

We will set aside the last column until later. The columns labelled 'Flow 1' and 'Flow 2' compare two flows consisting of 60 B and 1,500 B packets respectively. The body of the table considers two separate cases, one where the flows have an equal bit rate and the other with equal packet rates. In both cases, the two flows fill a 96 Mbps link. Therefore, in the equal bit rate case, they each have half the bit rate (48Mbps). Whereas, with equal packet rates, Flow 1 uses 25 times smaller packets so it gets 25 times less bit rate -- it only gets 1/(1+25) of the link capacity (96 Mbps / 26 = 4 Mbps after rounding). In contrast Flow 2 gets 25 times more bit rate (92 Mbps) in the equal packet rate case because its packets are 25 times larger. The packet rate shown for each flow could easily be derived once the bit rate was known by dividing the bit rate by packet size, as shown in the column labelled 'Formula'.

Parameter	Formula	Flow 1	Flow 2	Combined
Packet size	s/8	60 B	1,500 B	(Mix)
Packet size	s	480 b	12,000 b	(Mix)
Pkt loss probability	p	0.1%	0.1%	0.1%
EQUAL BIT RATE CASE				
Bit rate	x	48 Mbps	48 Mbps	96 Mbps
Packet rate	u = x/s	100 kpps	4 kpps	104 kpps
Absolute pkt-loss rate	p*u	100 pps	4 pps	104 pps
Absolute bit-loss rate	p*u*s	48 kbps	48 kbps	96 kbps
Ratio of lost/sent pkts	p*u/u	0.1%	0.1%	0.1%
Ratio of lost/sent bits	p*u*s/(u*s)	0.1%	0.1%	0.1%
EQUAL PACKET RATE CASE				
Bit rate	x	4 Mbps	92 Mbps	96 Mbps
Packet rate	u = x/s	8 kpps	8 kpps	15 kpps
Absolute pkt-loss rate	p*u	8 pps	8 pps	15 pps
Absolute bit-loss rate	p*u*s	4 kbps	92 kbps	96 kbps
Ratio of lost/sent pkts	p*u/u	0.1%	0.1%	0.1%
Ratio of lost/sent bits	p*u*s/(u*s)	0.1%	0.1%	0.1%

Table 5: Absolute Loss Rates and Loss Ratios for Flows of Small and Large Packets and Both Combined

So far, we have merely set up the scenarios. We now consider congestion notification in the scenario. Two TCP flows with the same round-trip time aim to equalise their packet-loss rates over time; that is, the number of packets lost in a second, which is the packets per second (u) multiplied by the probability that each one is dropped (p). Thus, TCP converges on the case labelled 'Equal packet rate' in the table, where both flows aim for the same absolute packet-loss rate (both 8 pps in the table).

Packet-mode drop actually gives flows sufficient information to measure their loss rate in bits per second, if they choose, not just packets per second. Each flow can count the size of a lost or marked packet and scale its rate response in proportion (as TFRC-SP does). The result is shown in the row entitled 'Absolute bit-loss rate', where the bits lost in a second is the packets per second (u) multiplied by the probability of losing a packet (p) multiplied by the packet size (s). Such an algorithm would try to remove any imbalance in the bit-loss rate such as the wide disparity in the case labelled 'Equal packet rate' (4k bps vs. 92 kbps). Instead, a packet-size-dependent algorithm would aim for equal bit-loss rates, which would drive both flows towards the case labelled 'Equal bit rate', by driving them to equal bit-loss rates (both 48 kbps in this example).

The explanation so far has assumed that each flow consists of packets of only one constant size. Nonetheless, it extends naturally to flows with mixed packet sizes. In the right-most column of Table 5, a flow of mixed-size packets is created simply by considering Flow 1 and Flow 2 as a single aggregated flow. There is no need for a flow to maintain an average packet size. It is only necessary for the transport to scale its response to each congestion indication by the size of each individual lost (or marked) packet. Taking, for example, the case labelled 'Equal packet rate', in one second about 8 small packets and 8 large packets are lost (making closer to 15 than 16 losses per second due to rounding). If the transport multiplies each loss by its size, in one second it responds to 8*480 and 8*12,000 lost bits, adding up to 96,000 lost bits in a second. This double checks correctly, being the same as 0.1% of the total bit rate of 96 Mbps. For completeness, the formula for absolute bit-loss rate is p(u1*s1+u2*s2).

Incidentally, a transport will always measure the loss probability the same, irrespective of whether it measures in packets or in bytes. In other words, the ratio of lost packets to sent packets will be the same as the ratio of lost bytes to sent bytes. (This is why TCP's bit rate is still proportional to packet size, even when byte counting is used, as recommended for TCP in [RFC5681], mainly for orthogonal security reasons.) This is intuitively obvious by comparing two example flows; one with 60 B packets, the other with 1,500 B packets. If both flows pass through a queue with drop probability 0.1%, each flow will lose 1 in 1,000 packets. In the stream of 60 B packets, the ratio of lost bytes to sent bytes will be 60 B in every 60,000 B; and in the stream of 1,500 B packets, the loss ratio will be 1,500 B out of 1,500,000 B. When the transport responds to the ratio of lost to sent packets, it will measure the same ratio whether it measures in packets or bytes: 0.1% in both cases. The fact that this ratio is the same whether measured in packets or bytes can be seen in Table 5, where the ratio of lost packets to sent packets and the ratio of lost bytes to sent bytes is always 0.1% in all cases (recall that the scenario was set up with p=0.1%).

This discussion of how the ratio can be measured in packets or bytes is only raised here to highlight that it is irrelevant to this memo! Whether or not a transport depends on packet size depends on how this ratio is used within the congestion control algorithm.

So far, we have shown that packet-mode drop passes sufficient information to the transport layer so that the transport can take bit congestion into account, by using the sizes of the packets that indicate congestion. We have also shown that the transport can

choose not to take packet size into account if it wishes. We will now consider whether the transport can know which to do.

B.2. Bit-Congestible and Packet-Congestible Indications

As a thought-experiment, imagine an idealised congestion notification protocol that supports both bit-congestible and packet-congestible resources. It would require at least two ECN flags, one for each of the bit-congestible and packet-congestible resources.

1. A packet-congestible resource trying to code congestion level p_p into a packet stream should mark the idealised 'packet congestion' field in each packet with probability p_p irrespective of the packet's size. The transport should then take a packet with the packet congestion field marked to mean just one mark, irrespective of the packet size.

2. A bit-congestible resource trying to code time-varying byte-congestion level p_b into a packet stream should mark the 'byte congestion' field in each packet with probability p_b, again irrespective of the packet's size. Unlike before, the transport should take a packet with the byte congestion field marked to count as a mark on each byte in the packet.

This hides a fundamental problem -- much more fundamental than whether we can magically create header space for yet another ECN flag, or whether it would work while being deployed incrementally. Distinguishing drop from delivery naturally provides just one implicit bit of congestion indication information -- the packet is either dropped or not. It is hard to drop a packet in two ways that are distinguishable remotely. This is a similar problem to that of distinguishing wireless transmission losses from congestive losses.

This problem would not be solved, even if ECN were universally deployed. A congestion notification protocol must survive a transition from low levels of congestion to high. Marking two states is feasible with explicit marking, but it is much harder if packets are dropped. Also, it will not always be cost-effective to implement AQM at every low-level resource, so drop will often have to suffice.

We are not saying two ECN fields will be needed (and we are not saying that somehow a resource should be able to drop a packet in one of two different ways so that the transport can distinguish which sort of drop it was!). These two congestion notification channels are a conceptual device to illustrate a dilemma we could face in the future. Section 3 gives four good reasons why it would be a bad idea to allow for packet size by biasing drop probability in favour of small packets within the network. The impracticality of our thought

experiment shows that it will be hard to give transports a practical way to know whether or not to take into account the size of congestion indication packets.

Fortunately, this dilemma is not pressing because by design most equipment becomes bit-congested before its packet processing becomes congested (as already outlined in Section 1.1). Therefore, transports can be designed on the relatively sound assumption that a congestion indication will usually imply bit congestion.

Nonetheless, although the above idealised protocol isn't intended for implementation, we do want to emphasise that research is needed to predict whether there are good reasons to believe that packet congestion might become more common, and if so, to find a way to somehow distinguish between bit and packet congestion [RFC3714].

Recently, the dual resource queue (DRQ) proposal [DRQ] has been made on the premise that, as network processors become more cost-effective, per-packet operations will become more complex (irrespective of whether more function in the network is desirable). Consequently the premise is that CPU congestion will become more common. DRQ is a proposed modification to the RED algorithm that folds both bit congestion and packet congestion into one signal (either loss or ECN).

Finally, we note one further complication. Strictly, packet-congestible resources are often cycle-congestible. For instance, for routing lookups, load depends on the complexity of each lookup and whether or not the pattern of arrivals is amenable to caching. This also reminds us that any solution must not require a forwarding engine to use excessive processor cycles in order to decide how to say it has no spare processor cycles.

Appendix C. Byte-Mode Drop Complicates Policing Congestion Response

This section is informative, not normative.

There are two main classes of approach to policing congestion response: (i) policing at each bottleneck link or (ii) policing at the edges of networks. Packet-mode drop in RED is compatible with either, while byte-mode drop precludes edge policing.

The simplicity of an edge policer relies on one dropped or marked packet being equivalent to another of the same size without having to know which link the drop or mark occurred at. However, the byte-mode drop algorithm has to depend on the local MTU of the line -- it needs to use some concept of a 'normal' packet size. Therefore, one dropped or marked packet from a byte-mode drop algorithm is not

necessarily equivalent to another from a different link. A policing function local to the link can know the local MTU where the congestion occurred. However, a policer at the edge of the network cannot, at least not without a lot of complexity.

The early research proposals for type (i) policing at a bottleneck link [pBox] used byte-mode drop, then detected flows that contributed disproportionately to the number of packets dropped. However, with no extra complexity, later proposals used packet-mode drop and looked for flows that contributed a disproportionate amount of dropped bytes [CHOKe_Var_Pkt].

Work is progressing on the Congestion Exposure (ConEx) protocol [RFC6789], which enables a type (ii) edge policer located at a user's attachment point. The idea is to be able to take an integrated view of the effect of all a user's traffic on any link in the internetwork. However, byte-mode drop would effectively preclude such edge policing because of the MTU issue above.

Indeed, making drop probability depend on the size of the packets that bits happen to be divided into would simply encourage the bits to be divided into smaller packets in order to confuse policing. In contrast, as long as a dropped/marked packet is taken to mean that all the bytes in the packet are dropped/marked, a policer can remain robust against sequences of bits being re-divided into different size packets or across different size flows [Rate_fair_Dis].

Authors' Addresses

Bob Briscoe
BT
B54/77, Adastral Park
Martlesham Heath
Ipswich IP5 3RE
UK

Phone: +44 1473 645196
EMail: bob.briscoe@bt.com
URI: http://bobbriscoe.net/

Jukka Manner
Aalto University
Department of Communications and Networking (Comnet)
P.O. Box 13000
FIN-00076 Aalto
Finland

Phone: +358 9 470 22481
EMail: jukka.manner@aalto.fi
URI: http://www.netlab.tkk.fi/~jmanner/