Transport Area Working Group
Internet-Draft
Intended status: Informational
Expires: August 27, 2008

B. Briscoe
BT & UCL

February 24, 2008

Byte and Packet Congestion Notification
draft-briscoe-tsvwg-byte-pkt-mark-02

Status of this Memo

Copyright Notice

Abstract

This memo concerns dropping or marking packets using active queue management (AQM) such as random early detection (RED) or pre-congestion notification (PCN). The primary conclusion is that packet size should be taken into account when transports decode congestion indications, not when network equipment writes them. Reducing drop of small packets has some tempting advantages: i) it drops less 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 specific transport problems. Network layer algorithms like the byte-mode packet drop variant of RED 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.

Table of Contents

Changes from Previous Versions

To be removed by the RFC Editor on publication.

Full incremental diffs between each version are available at <http://www.cs.ucl.ac.uk/staff/B.Briscoe/pubs.html#byte-pkt-mark> (courtesy of the rfcdiff tool):

From -01 to -02 (this version):

Abstract reorganised to align with clearer separation of issue in the memo.

Introduction reorganised with motivating arguments removed to new Section 2.

Clarified avoiding lock-out of large packets is not the main or only motivation for RED.

Mentioned choice of drop or marking explicitly throughout, rather than trying to coin a word to mean either.

Generalised the discussion throughout to any packet forwarding function on any network equipment, not just routers.

Clarified the last point about why this is a good time to sort out this issue: because it will be hard / impossible to design new transports unless we decide whether the network or the transport is allowing for packet size.

Added statement explaining the horizon of the memo is long term, but with short term expediency in mind.

Added material on scaling congestion control with packet size (Section 2.1).

Separated out issue of normalising TCP's bit rate from issue of preference to control packets (Section 2.3).

Divided up Congestion Measurement section for clarity, including new material on fixed size packet buffers and buffer carving (Section 4.1.1 & Section 6.2.1) and on congestion measurement in wireless link technologies without queues (Section 4.2).

Added section on 'Making Transports Robust against Control Packet Losses' (Section 6.2.3) with existing & new material included.

Added tabulated results of vendor survey on byte-mode drop variant of RED (Table 2).

From -00 to -01:

Clarified applicability to drop as well as ECN.

Highlighted DoS vulnerability.

Emphasised that drop-tail suffers from similar problems to byte-mode drop, so only byte-mode drop should be turned off, not RED itself.

Clarified the original apparent motivations for recommending byte-mode drop included protecting SYNs and pure ACKs more than equalising the bit rates of TCPs with different segment sizes. Removed some conjectured motivations.

Added support for updates to TCP in progress (ackcc & ecn-syn-ack).

Updated survey results with newly arrived data.

Pulled all recommendations together into the conclusions.

Moved some detailed points into two additional appendices and a note.

Considerable clarifications throughout.

Updated references

Requirements notation

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].

1. Introduction

When notifying 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 drop-tail queues. This memo aims to state the principles we should be using and to come to conclusions on what these principles will mean for future protocol design, taking into account the deployments we have already.

Note that the byte vs. packet dilemma concerns congestion notification irrespective of whether it is signalled implicitly by drop or using explicit congestion notification (ECN [RFC3168] or PCN [I-D.ietf-pcn-architecture]). 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.

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, 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).

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.

Resources may be congestible at higher levels of granularity than 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 applied for congestion notification protocols controlling per-flow and per-session processing or state.

The byte vs. packet dilemma arises at three stages in the congestion notification process:

Measuring congestion When the congested resource decides locally how to measure how congested it is. (Should the queue be measured in bytes or packets?);

Coding congestion notification into the wire protocol: When the congested resource decides how to notify the level of congestion. (Should the level of notification depend on the byte-size of each particular packet carrying the notification?);

Decoding congestion notification from the wire protocol: When the transport interprets the notification. (Should the byte-size of a missing or marked packet be taken into account?).

In RED, whether to use packets or bytes when measuring queues is called packet-mode or byte-mode queue measurement. This choice is now fairly well understood but is included in Section 4 to document it in the RFC series.

The controversy is mainly around the other two stages: whether to allow for packet size when the network codes or when the transport decodes congestion notification. In RED, the variant that reduces drop probability for packets based on their size in bytes is called byte-mode drop, while the variant that doesn't is called packet mode drop. Whether queues are measured in bytes or packets is an orthogonal choice, termed byte-mode queue measurement or packet-mode queue measurement.

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]. However, all the implementers who responded to our survey have not followed this advice. The primary purpose of this memo is to build a definitive consensus against deliberate preferential treatment for small packets in AQM algorithms and to record this advice within the RFC series.

Now is a good time to discuss whether fairness between different sized packets would best be implemented in the network layer, or at the transport, for a number of reasons:

1. The packet vs. byte issue requires speedy resolution because the IETF pre-congestion notification (PCN) working group has been chartered to produce a standards track specification of its congestion notification (AQM) algorithm [PCNcharter];

2. [RFC2309] says RED may either take account of packet size or not when dropping, but gives no recommendation between the two, referring instead to advice on the performance implications in an

email [pktByteEmail], which recommends byte-mode drop. Further, just before RFC2309 was issued, an addendum was added to the archived email that revisited the issue of packet vs. byte-mode drop in its last para, making the recommendation less clear-cut;

3. Without the present memo, the only advice in the RFC series on packet size bias in AQM algorithms would be a reference to an archived email in [RFC2309] (including an addendum at the end of the email to correct the original).

4. The IRTF Internet Congestion Control Research Group (ICCRG) recently took on the challenge of building consensus on what common congestion control support should be required from network forwarding functions in future [I-D.irtf-iccrg-welzl-congestion-control-open-research]. The wider Internet community needs to discuss whether the complexity of adjusting for packet size should be in the network or in transports;

5. Given there are many good reasons why larger path max transmission units (PMTUs) would help solve a number of scaling issues, we don't want to create any bias against large packets that is greater than their true cost;

6. The IETF has started to consider the question of fairness between flows that use different packet sizes (e.g. in the small-packet variant of TCP-friendly rate control, TFRC-SP [RFC4828]). Given transports with different packet sizes, if we don't decide whether the network or the transport should allow for packet size, it will be hard if not impossible to design any transport protocol so that its bit-rate relative to other transports meets design guidelines [RFC5033] (Note however that, if the concern were fairness between users, rather than between flows [Rate_fair_Dis], relative rates between flows would have to come under run-time control rather than being embedded in protocol designs).

This memo is initially concerned with how we should correctly scale congestion control functions with packet size for the long term. But 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. It turns out that the 'correct' variant of RED to deploy seems to be the one everyone has deployed, and no-one who responded to our survey has implemented the other variant. However, at the transport layer, TCP congestion control is a widely deployed protocol that we argue doesn't scale correctly with packet size. To date this hasn't been a significant problem because most TCPs have been used with similar packet sizes. But, as we design new congestion

controls, we should build in scaling with packet size rather than assuming we should follow TCP's example.

Motivating arguments for our advice are given next in Section 2. Then the body of the memo starts from first principles, defining congestion notification in Section 3 then determining the correct way to measure congestion (Section 4) and to design an idealised congestion notification protocol (Section 5). It then surveys the advice given previously in the RFC series, the research literature and the deployed legacy (Section 6) before listing outstanding issues (Section 7) that will need resolution both to achieve the ideal protocol and to handle legacy. After discussing security considerations (Section 8) strong recommendations for the way forward are given in the conclusions (Section 9).

2. Motivating Arguments

2.1. Scaling Congestion Control with Packet Size

There are two ways of interpreting a dropped or marked packet. It can either be considered as a single loss event or as loss/marking of the bytes in the packet. Here we try to design a test to see which approach scales with packet size.

Imagine a bit-congestible link shared by many flows, so that each busy period tends to cause packets to be lost from different flows. The test compares two identical scenarios with the same applications, the same numbers of sources and the same load. But the sources break the load into large packets in one scenario and small packets in the other. Of course, because the load is the same, there will be proportionately more packets in the small packet case.

The test of whether a congestion control scales with packet size is that it should respond in the same way to the same congestion excursion, irrespective of the size of the packets that the bytes causing congestion happen to be broken down into.

A bit-congestible queue suffering a congestion excursion has to drop or mark the same excess bytes whether they are in a few large packets or many small packets. So for the same congestion excursion, the same amount of bytes have to be shed to get the load back to its operating point. But, of course, for smaller packets more packets will have to be discarded to shed the same bytes.

If all the transports interpret each drop/mark as a single loss event irrespective of the size of the packet dropped, they will respond more to the same congestion excursion, failing our test. On the

other hand, if they respond proportionately less when smaller packets are dropped/marked, overall they will be able to respond the same to the same congestion excursion.

Therefore, for a congestion control to scale with packet size it should respond to dropped or marked bytes (as TFRC-SP [RFC4828] effectively does), not just to dropped or marked packets irrespective of packet size (as TCP does).

The email [pktByteEmail] referred to by RFC2309 says the question of whether a packet's own size should affect its drop probability "depends on the dominant end-to-end congestion control mechanisms". But we argue the network layer should not be optimised for whatever transport is predominant.

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. But even though reducing the drop probability of small packets helps ensure TCPs with different packet sizes will achieve similar bit rates, we argue this should be achieved in TCP itself, not in the network.

Effectively, favouring small packets is reverse engineering of the network layer around 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?"

2.2. 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]. Again, imagine a scenario where the same bit rate of packets will contribute the same to 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 this case:

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 traffic in large packets, 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. The byte-mode drop variant of RED amplifies small packet attacks.

Drop-tail queues amplify small packet attacks even more than RED byte-mode drop (see the Security Considerations section Section 8). Wherever possible neither should be used.

Normal transports: Even if a transport is not malicious, if it finds small packets go faster, it will tend to act in its own interest and use them. Queues that give advantage to small packets create an evolutionary pressure for 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 two flows arrive at a bit-congestible transmission link each with the same bit rate, say 1Mbps, but one consists of 1500B and the other 60B 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, 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 [RFC4828](S.B.4)[Note_Variation]). Even though both flows arrive with the same bit rate, the bit rate the RED queue aims to pass to the line will be 750k for the flow of larger packet but 990k for the smaller packets (but because of rate variation it will be less than this target). It can be seen that this behaviour reopens the same denial of service vulnerability that drop tail queues offer to floods of small packet, though not necessarily as strongly (see Section 8).

2.3. Small != Control

It is tempting to drop small packets with lower probability to improve performance, because many control packets are small (TCP SYNs & ACKs, DNS queries & responses, SIP messages, HTTP GETs, etc) and dropping fewer control packets considerably improves performance. 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 again, rather than fix these problems in the network layer, we argue that the transport should be made more robust against losses of

control packets (see 'Making Transports Robust against Control Packet Losses' in Section 6.2.3).

3. Working Definition of Congestion Notification

Rather than aim to achieve what many have tried and failed, this memo will not try to define congestion. It will give a working definition of what congestion notification should be taken to mean for this document. Congestion notification is a changing signal that aims to communicate the ratio E/L, where E is the instantaneous excess load offered to a resource that it cannot (or would not) serve and L is the instantaneous offered load.

The phrase `would not serve' is added, because AQM systems (e.g. RED, PCN [I-D.ietf-pcn-architecture]) use a virtual capacity smaller than actual capacity, then notify congestion of this virtual capacity in order to avoid congestion of the actual capacity.

Note that the denominator is offered load, not capacity. Therefore congestion notification is a real number bounded by the range [0,1]. This ties in with the most well-understood form of congestion notification: drop rate. It also means that congestion has a natural interpretation as a probability; the probability of offered traffic not being served (or being marked as at risk of not being served). Appendix B describes a further incidental benefit that arises from using load as the denominator of congestion notification.

4. Congestion Measurement

4.1. Congestion Measurement by Queue Length

Queue length is usually the most correct and simplest way to measure congestion of a resource. To avoid the pathological effects of drop tail, an AQM function can then be used to transform queue length into the probability of dropping or marking a packet (e.g. RED's piecewise linear function between thresholds). If the resource is bit-congestible, the length of the queue SHOULD be measured in bytes. If the resource is packet-congestible, the length of the queue SHOULD be measured in packets. 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. We discuss the implications on RED's byte mode and packet mode for measuring queue length in Section 6.

4.1.1. Fixed Size Packet Buffers

Some, mostly older, queuing hardware sets aside fixed sized buffers in which to store each packet in the queue. Also, with some hardware, any fixed sized buffers not completely filled by a packet are padded when transmitted to the wire. If we imagine a theoretical forwarding system with both queuing and transmission in fixed, MTU-sized units, it should clearly be treated as packet-congestible, because the queue length in packets would be a good model of congestion of the lower layer link.

If we now imagine a hybrid forwarding system with transmission delay largely dependent on the byte-size of packets but buffers of one MTU per packet, it should strictly require a more complex algorithm to determine the probability of congestion. It should be treated as two resources in sequence, where the sum of the byte-sizes of the packets within each packet buffer models congestion of the line while the length of the queue in packets models congestion of the queue. Then the probability of congesting the forwarding buffer would be a conditional probability--conditional on the previously calculated probability of congesting the line.

However, in systems that use fixed size buffers, it is unusual for all the buffers used by an interface to be the same 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, it is easier to work out what should be done if we temporarily set aside the possibility of such borrowing. Then, with fixed pools of buffers for different sized packets and no borrowing, the size of each pool and the current queue length in each pool would both be measured in packets. So an AQM algorithm would have to maintain the queue length for each pool, and judge whether to drop/mark a packet of a particular size by looking at the pool for packets of that size and using the length (in packets) of its queue.

We now return to the issue we temporarily set aside: small packets borrowing space in larger buffers. In this case, the only difference is that the pools for smaller packets have a maximum queue size that includes all the pools for larger packets. And every time a packet takes a larger buffer, the current queue size has to be incremented for all queues in the pools of buffers less than or equal to the buffer size used.

We will return to borrowing of fixed sized buffers when we discuss biasing the drop/marking probability of a specific packet because of

its size in Section 6.2.1. But here we can give a simple summary of the present discussion on how to measure the length of queues of fixed buffers: no matter how complicated the scheme is, ultimately any fixed buffer system will need to measure its queue length in packets not bytes.

4.2. 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, wireless spectrum is bit-congestible (for a given coding scheme), because interference increases with the rate at which bits are transmitted. But wireless link protocols do not always maintain a queue that depends on spectrum interference. Similarly, power limited resources are also 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.

However, AQM algorithms don't require a queue in order to work. For instance spectrum congestion can be modelled by signal quality using 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.

5. Idealised Wire Protocol Coding

We will start by inventing an idealised congestion notification protocol before discussing how to make it practical. The idealised protocol is shown to be correct using examples in Appendix A. Congestion notification involves the congested resource coding a congestion notification signal into the packet stream and the transports decoding it. The idealised protocol uses two different fields in each datagram to signal congestion: one for byte congestion and one for packet congestion.

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 just a conceptual device. They allow us to defer having to

decide whether to distinguish between byte and packet congestion when the network resource codes the signal or when the transport decodes it.

However, although this idealised mechanism isn't intended for implementation, we do want to emphasise that we may need to find a way to implement it, because it could become necessary to somehow distinguish between bit and packet congestion [RFC3714]. Currently a design goal of network processing equipment such as routers and firewalls is to keep packet processing uncongested even under worst case bit rates with minimum packet sizes. Therefore, packet-congestion is currently rare, but there is no guarantee that it will not become common with future technology trends.

The idealised wire protocol is given below. It accounts for packet sizes at the transport layer, not in the network, and then only in the case of bit-congestible resources. This avoids the perverse incentive to send smaller packets and the DoS vulnerability that would otherwise result if the network were to bias towards them (see the motivating argument about avoiding perverse incentives in Section 2.2). Incidentally, it also ensures 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:

- 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.

- 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.

The worked examples in Appendix A show that transports can extract sufficient and correct congestion notification from these protocols for cases when two flows with different packet sizes have matching bit rates or matching packet rates. Examples are also given that mix these two flows into one to show that a flow with mixed packet sizes would still be able to extract sufficient and correct information.

Sufficient and correct congestion information means that there is

sufficient information for the two different types of transport requirements:

Ratio-based: Established transport congestion controls like TCP's [RFC2581] aim to achieve equal segment rates per RTT through the same bottleneck--TCP friendliness [RFC3448]. They work with the ratio of dropped to delivered segments (or marked to unmarked segments in the case of ECN). The example scenarios show that these ratio-based transports are effectively the same whether counting in bytes or packets, because the units cancel out. (Incidentally, this is why TCP's bit rate is still proportional to packet size even when byte-counting is used, as recommended for TCP in [I-D.ietf-tcpm-rfc2581bis], mainly for orthogonal security reasons.)

Absolute-target-based: Other congestion controls proposed in the research community aim to limit the volume of congestion caused to a constant weight parameter. [MulTCP][WindowPropFair] are examples of weighted proportionally fair transports designed for cost-fair environments [Rate_fair_Dis]. In this case, the transport requires a count (not a ratio) of dropped/marked bytes in the bit-congestible case and of dropped/marked packets in the packet congestible case.

6. The State of the Art

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. This email clarified that there were in fact two orthogonal choices: whether to measure queue length in bytes or packets (Section 6.1 below) and whether the drop probability of an individual packet should depend on its own size (Section 6.2 below).

6.1. Congestion Measurement: Status

The choice of which metric to use to measure queue length was left open in RFC2309. 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 (see Section 4).

Where buffers are not configured or legacy buffers cannot be configured to the above guideline, we don't have to make allowances for such legacy in future protocol design. If a bit-congestible buffer is measured in packets, the operator will have set the thresholds mindful of a typical mix of packets sizes. Any AQM algorithm on such a buffer will be oversensitive to high proportions of small packets, e.g. a DoS attack, and undersensitive to high proportions of large packets. But an operator can safely keep such a legacy buffer because any undersensitivity during unusual traffic mixes cannot lead to congestion collapse given the buffer will eventually revert to tail drop, discarding proportionately more large packets.

Some modern queue implementations give a choice for setting RED's thresholds in byte-mode or packet-mode. This may merely be an administrator-interface preference, not altering how the queue itself is measured but on some hardware it does actually change the way it measures its queue. Whether a resource is bit-congestible or packet-congestible is a property of the resource, so an admin SHOULD NOT ever need to, or be able to, configure the way a queue measures itself.

We believe the question of whether to measure queues in bytes or packets is fairly well understood these days. The only outstanding issues concern how to measure congestion when the queue is bit congestible but the resource is packet congestible or vice versa (see Section 4). But there is no controversy over what should be done. It's just you have to be an expert in probability to work out what should be done and, even if you have, it's not always easy to find a practical algorithm to implement it.

6.2. Congestion Coding: Status

6.2.1. Network Bias when Encoding

The previously mentioned email [pktByteEmail] referred to by [RFC2309] said that the choice over whether a packet's own size should affect its drop probability "depends on the dominant end-to-end congestion control mechanisms". [Section 2 argues against this approach, citing the excellent advice in RFC3246.] The referenced

email went on to argue 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, but advised that most scarce resources in the Internet were currently bit-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]. [The problem could have been solved by making the policing mechanism count the volume of bytes randomly dropped, not the number of packets.]

A few months before RFC2309 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 byte-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 drop rate dependent on the square of relative packet size. This was indeed consistent with one stated 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.

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 for this change in the literature, however Eddy and Allman conducted experiments [REDbias] that assessed how sensitive RED was to this parameter, amongst other things. No-one seems to have pointed out that this changed algorithm can often lead to drop probabilities of greater than 1 [which should ring alarm bells hinting that there's 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.

The byte-mode drop variant of RED is, of course, not the only possible bias towards small packets in queueing algorithms. We have already mentioned that tail-drop queues naturally tend to lock-out large packets once they are full. But also queues with fixed sized 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. As was explained in Section 4.1.1 on fixed size buffer carving, 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.

However, in itself, the bias towards small packets caused by buffer borrowing is perfectly correct. Lower drop probability for small packets is legitimate in buffer borrowing schemes, because small packets genuinely congest the machine's buffer memory less than large packets, given they can fit in more spaces. The bias towards small packets is not artificially added (as it is in RED's byte-mode drop algorithm), it merely reflects the reality of the way fixed buffer memory gets congested. 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 a good AQM algorithm like RED with packet-mode drop should be used with fixed buffer memories where possible. If RED is too complicated to implement with multiple fixed buffer pools, the minimum necessary to prevent large packet lock-out is to ensure smaller packets never use the last available buffer in any of the pools for larger packets.

6.2.2. Transport Bias when Decoding

The above proposals to alter the network layer to give a bias towards smaller packets have largely carried on outside the IETF process (unless one counts a reference in an informational RFC to an archived email!). 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 account of 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 account of packet sizes relative to typical TCP packet sizes. This proposes a new small-packet variant of TCP-friendly rate control [RFC3448] called TFRC-SP [RFC4828]. Essentially, it proposes a rate equation that inflates the flow rate by the ratio of a typical TCP segment size (1500B 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 this TFRC-SP spec 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 it didn't sufficiently bias towards small packets to make TCP independent of packet size.

So the TFRC-SP spec doesn't address the issue of which of the network or the transport _should_ handle fairness between different packet sizes. In its Appendix B.4 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 Section 6.2.4). Incidentally, VP-TFRC also proposed that byte-mode RED dropping should really square the packet size compensation factor (like that of RED_5, but apparently unaware of it).

Pre-congestion notification [I-D.ietf-pcn-architecture] is a proposal 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 proposed PCN marking algorithms have been designed not to take account of packet size when forwarding through queues. Instead the general principle has been to take account of the sizes of marked packets when monitoring the fraction of marking at the edge of the network.

6.2.3. Making Transports Robust against Control Packet Losses

Recently, two drafts have proposed changes to TCP that make it more robust against losing small control packets [I-D.ietf-tcpm-ecnsyn] [I-D.floyd-tcpm-ackcc]. In both cases they note that the case for

these 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 no proposals exist as far as we know, it would also be possible and perfectly valid to make control packets robust against drop by explicitly requesting a lower drop probability using their Diffserv code point [RFC2474] to request a scheduling class with lower drop.

The re-ECN protocol proposal [Re-TCP] is designed so that transports can be made more robust against losing control packets. It gives queues an incentive to optionally give preference against drop to packets with the 'feedback not established' codepoint in the proposed 'extended ECN' field. Senders have incentives to use this codepoint sparingly, but they can use it on control packets to reduce their chance of being dropped. For instance, the proposed modification to TCP for re-ECN uses this codepoint on the SYN and SYN-ACK.

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 efforts Internet at minimal cost.

6.2.4. Congestion Coding: Summary of Status

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(sp)	1/(s.sqrt(p))

Table 1: Dependence of flow bit-rate per RTT on packet size s and drop rate p when network and/or transport bias towards small packets to varying degrees

Table 1 aims to summarise the positions we may now be in. 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 [RFC2581] and TFRC [RFC3448] on the top row with TFRC-SP [RFC4828] below. Suppressing all inessential details the table shows that independence from packet size should either be achievable by not altering the TCP transport in a RED_5 network, or using the small packet TFRC-SP transport 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 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 selection of congested queues, each path will present a different hybrid scenario to its transport.

Whatever, we can see that the linear byte-mode drop column in the middle considerably complicates the Internet. It's a half-way house that doesn't bias enough towards small packets even if one believes the network should be doing the biasing. We argue below that _all_ network layer bias 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).

A survey has been conducted of 84 vendors to assess how widely drop probability based on packet size has been implemented in RED. Prior to the survey, an individual approach to Cisco received confirmation that, having checked the code-base for each of the product ranges, Cisco has not implemented any discrimination based on packet size in any AQM algorithm in any of its products. Also an individual approach to Alcatel-Lucent drew a confirmation that it was very likely that none of their products contained RED code that implemented any packet-size bias.

Turning to our more formal survey (Table 2), about 19% of those surveyed have replied so far, giving a sample size of 16. Although we do not have permission to identify the respondents, we can say that those that have responded include most of the larger vendors, covering a large fraction of the market. They 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. So far, all those who have responded have confirmed that they have not implemented the variant of RED with drop dependent on packet size (2 are fairly sure they haven't but need to check more thoroughly).

Response	No. of vendors	%age 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 2: Vendor Survey on byte-mode drop variant of RED (lower drop probability for small packets)

Where reasons have been given, the extra complexity of packet bias code has been most prevalent, though one vendor had a more principled reason for avoiding it--similar to the argument of this document. We have established that Linux does not implement RED with packet size drop bias, although we have not investigated a wider range of open source code.

Finally, we repeat that RED's byte mode drop is not the only way to bias towards small packets--tail-drop tends to lock-out large packets very effectively. 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. My own company (BT) has widely deployed RED, but there are bound to be many tail-drop queues, 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 6.2.1, 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.

7. Outstanding Issues and Next Steps

7.1. Bit-congestible World

For a connectionless network with only bit-congestible resources we believe the recommended position is now unarguably clear--that the network should not make allowance for packet sizes and the transport should. This leaves two outstanding issues:

- How to handle any legacy of AQM with byte-mode drop already deployed;

- The need to start a programme to update transport congestion control protocol standards to take account of packet size.

The sample of returns from our vendor survey Section 6.2.4 suggest that byte-mode packet drop seems not to be implemented at all let alone deployed, or if it is, it is likely to be very sparse. Therefore, we do not really need a migration strategy from all but nothing to nothing.

A programme of standards updates to take account of packet size in transport congestion control protocols has started with TFRC-SP [RFC4828], while weighted TCPs implemented in the research community [WindowPropFair] could form the basis of a future change to TCP congestion control [RFC2581] itself.

7.2. Bit- & Packet-congestible World

Nonetheless, a connectionless network with both bit-congestible and packet-congestible resources is a different matter. If we believe we should allow for this possibility in the future, this space contains a truly open research issue.

The idealised wire protocol coding described in Section 5 requires at least two flags for congestion of bit-congestible and packet-congestible resources. This hides a fundamental problem--much more fundamental than whether we can magically create header space for yet another ECN flag in IPv4, or whether it would work while being deployed incrementally. A congestion notification protocol must survive a transition from low levels of congestion to high. Marking two states is feasible with explicit marking, but 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. Distinguishing drop from delivery naturally provides just one congestion flag--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.

We should also note that, strictly, packet-congestible resources are actually cycle-congestible because load also depends on the complexity of each look-up and whether the pattern of arrivals is amenable to caching or not. Further, this 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.

The problem of signalling packet processing congestion is not pressing, as most if not all Internet resources are designed to be bit-congestible before packet processing starts to congest. However,

given the IRTF 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 [I-D.irtf-iccrg-welzl-congestion-control-open-research], we must not give this problem no thought at all, just because it is hard and currently hypothetical.

8. Security Considerations

This draft recommends that queues do not bias drop probability towards small packets as this 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 drop-tail queues gave to small packets. Of course, if transports really want to make the greatest gains, they don't have to respond to congestion anyway. But we don't want applications that are trying to behave to discover that they can go faster by using smaller packets.

In practice, transports cannot all be trusted to respond to congestion. So another reason for recommending that queues do 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 drop-tail'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 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.

9. Conclusions

The strong conclusion is that AQM algorithms such as RED SHOULD NOT use byte-mode drop. More generally, the Internet's congestion notification protocols (drop, ECN & PCN) SHOULD take account of packet size when the notification is read by the transport layer, NOT when it is written by the network layer. This approach offers sufficient and correct 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.

The alternative of deflating RED's drop probability for smaller packet sizes (byte-mode drop) has no enduring advantages. It is more complex, it creates the perverse incentive to fragment segments into tiny pieces and it reopens the vulnerability to floods of small-packets that drop-tail queues suffered from and AQM was designed to remove. Byte-mode drop is a change to the network layer that makes allowance for an omission from the design of TCP, effectively reverse engineering the network layer to contrive to make two TCPs with different packet sizes run at equal bit rates (rather than packet rates) under the same path conditions. It also improves TCP performance by reducing the chance that a SYN or a pure ACK will be dropped, because they are small. But we SHOULD NOT hack the network layer to improve or fix certain transport protocols. No matter how predominant a transport protocol is (even if it's TCP), trying to correct for its failings by biasing towards small packets in the network layer creates a perverse incentive to break down all flows from all transports into tiny segments.

So far, our survey of 84 vendors across the industry has drawn responses from about 19%, none of whom have implemented the byte mode packet drop variant of RED. Given there appears to be little, if any, installed base recommending removal of byte-mode drop from RED is possibly only a paper exercise with few, if any, incremental deployment issues.

If a vendor has implemented byte-mode drop, and an operator has turned it on, it is strongly RECOMMENDED that it SHOULD be turned off. Note that RED as a whole SHOULD NOT be turned off, as without it, a drop tail queue also biases against large packets. But note also that turning off byte-mode may alter the relative performance of applications using different packet sizes, so it would be advisable to establish the implications before turning it off.

Instead, the IETF transport area should continue its programme of updating congestion control protocols to take account of packet size and to make transports less sensitive to losing control packets like

SYNs and pure ACKS.

NOTE WELL that RED's byte-mode queue measurement is fine, being completely orthogonal to byte-mode drop. 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.

The above conclusions cater for the Internet as it is today with most, if not all, resources being primarily bit-congestible. A secondary conclusion of this memo is that we may see more packet-congestible resources in the future, so research may be needed to extend the Internet's congestion notification (drop or ECN) so that it can handle a mix of bit-congestible and packet-congestible resources.

10. Acknowledgements

Thank you to Sally Floyd, who gave extensive and useful review comments. Also thanks for the reviews from Toby Moncaster and Arnaud Jacquet. I am 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.

11. Comments Solicited

Comments and questions are encouraged and very welcome. They can be addressed to the IETF Transport Area working group mailing list <tsvwg@ietf.org>, and/or to the authors.

Editorial Comments

[Note_Variation] The algorithm of the byte-mode drop variant of RED switches off any bias towards small packets whenever the smoothed queue length dictates that the drop probability of large packets should be 100%. In the example in the Introduction, as the large packet drop probability varies around 25% the small packet drop probability will vary around 1%, but with occasional jumps to 100% whenever the instantaneous queue (after drop) manages to sustain a length above the 100% drop point for longer than

the queue averaging period.

Appendix A. Example Scenarios

A.1. Notation

To prove the two sets of assertions in the idealised wire protocol (Section 5) are true, we will compare two flows with different packet sizes, s_1 and s_2 [bit/pkt], to make sure their transports each see the correct congestion notification. Initially, within each flow we will take all packets as having equal sizes, but later we will generalise to flows within which packet sizes vary. A flow's bit rate, x [bit/s], is related to its packet rate, u [pkt/s], by

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

We will consider a 2x2 matrix of four scenarios:

resource type and congestion level	A) Equal bit rates	B) Equal pkt rates
i) bit-congestible, p_b	(Ai)	(Bi)
ii) pkt-congestible, p_p	(Aii)	(Bii)

Table 3

A.2. Bit-congestible resource, equal bit rates (Ai)

Starting with the bit-congestible scenario, for two flows to maintain equal bit rates (Ai) the ratio of the packet rates must be the inverse of the ratio of packet sizes: u_2/u_1 = s_1/s_2. So, for instance, a flow of 60B packets would have to send 25x more packets to achieve the same bit rate as a flow of 1500B packets. If a congested resource marks proportion p_b of packets irrespective of size, the ratio of marked packets received by each transport will still be the same as the ratio of their packet rates, p_b.u_2/p_b.u_1 = s_1/s_2. So of the 25x more 60B packets sent, 25x more will be marked than in the 1500B packet flow, but 25x more won't be marked too.

In this scenario, the resource is bit-congestible, so it always uses our idealised bit-congestion field when it marks packets. Therefore the transport should count marked bytes not packets. But it doesn't actually matter for ratio-based transports like TCP (Section 5). The ratio of marked to unmarked bytes seen by each flow will be p_b, as

will the ratio of marked to unmarked packets. Because they are ratios, the units cancel out.

If a flow sent an inconsistent mixture of packet sizes, we have said it should count the ratio of marked and unmarked bytes not packets in order to correctly decode the level of congestion. But actually, if all it is trying to do is decode p_b, it still doesn't matter. For instance, imagine the two equal bit rate flows were actually one flow at twice the bit rate sending a mixture of one 1500B packet for every thirty 60B packets. 25x more small packets will be marked and 25x more will be unmarked. The transport can still calculate p_b whether it uses bytes or packets for the ratio. In general, for any algorithm which works on a ratio of marks to non-marks, either bytes or packets can be counted interchangeably, because the choice cancels out in the ratio calculation.

However, where an absolute target rather than relative volume of congestion caused is important (Section 5), as it is for congestion accountability [Rate_fair_Dis], the transport must count marked bytes not packets, in this bit-congestible case. Aside from the goal of congestion accountability, this is how the bit rate of a transport can be made independent of packet size; by ensuring the rate of congestion caused is kept to a constant weight [WindowPropFair], rather than merely responding to the ratio of marked and unmarked bytes.

Note the unit of byte-congestion volume is the byte.

A.3. Bit-congestible resource, equal packet rates (Bi)

If two flows send different packet sizes but at the same packet rate, their bit rates will be in the same ratio as their packet sizes, x_2/x_1 = s_2/s_1. For instance, a flow sending 1500B packets at the same packet rate as another sending 60B packets will be sending at 25x greater bit rate. In this case, if a congested resource marks proportion p_b of packets irrespective of size, the ratio of packets received with the byte-congestion field marked by each transport will be the same, p_b.u_2/p_b.u_1 = 1.

Because the byte-congestion field is marked, the transport should count marked bytes not packets. But because each flow sends consistently sized packets it still doesn't matter for ratio-based transports. The ratio of marked to unmarked bytes seen by each flow will be p_b, as will the ratio of marked to unmarked packets. Therefore, if the congestion control algorithm is only concerned with the ratio of marked to unmarked packets (as is TCP), both flows will be able to decode p_b correctly whether they count packets or bytes.

But if the absolute volume of congestion is important, e.g. for congestion accountability, the transport must count marked bytes not packets. Then the lower bit rate flow using smaller packets will rightly be perceived as causing less byte-congestion even though its packet rate is the same.

If the two flows are mixed into one, of bit rate x1+x2, with equal packet rates of each size packet, the ratio p_b will still be measurable by counting the ratio of marked to unmarked bytes (or packets because the ratio cancels out the units). However, if the absolute volume of congestion is required, the transport must count the sum of congestion marked bytes, which indeed gives a correct measure of the rate of byte-congestion p_b(x_1 + x_2) caused by the combined bit rate.

A.4. Pkt-congestible resource, equal bit rates (Aii)

Moving to the case of packet-congestible resources, we now take two flows that send different packet sizes at the same bit rate, but this time the pkt-congestion field is marked by the resource with probability p_p. As in scenario Ai with the same bit rates but a bit-congestible resource, the flow with smaller packets will have a higher packet rate, so more packets will be both marked and unmarked, but in the same proportion.

This time, the transport should only count marks without taking into account packet sizes. Transports will get the same result, p_p, by decoding the ratio of marked to unmarked packets in either flow.

If one flow imitates the two flows but merged together, the bit rate will double with more small packets than large. The ratio of marked to unmarked packets will still be p_p. But if the absolute number of pkt-congestion marked packets is counted it will accumulate at the combined packet rate times the marking probability, p_p(u_1+u_2), 26x faster than packet congestion accumulates in the single 1500B packet flow of our example, as required.

But if the transport is interested in the absolute number of packet congestion, it should just count how many marked packets arrive. For instance, a flow sending 60B packets will see 25x more marked packets than one sending 1500B packets at the same bit rate, because it is sending more packets through a packet-congestible resource.

Note the unit of packet congestion is packets.

A.5. Pkt-congestible resource, equal packet rates (Bii)

Finally, if two flows with the same packet rate, pass through a packet-congestible resource, they will both suffer the same proportion of marking, p_p, irrespective of their packet sizes. On detecting that the pkt-congestion field is marked, the transport should count packets, and it will be able to extract the ratio p_p of marked to unmarked packets from both flows, irrespective of packet sizes.

Even if the transport is monitoring the absolute amount of packets congestion over a period, still it will see the same amount of packet congestion from either flow.

And if the two equal packet rates of different size packets are mixed together in one flow, the packet rate will double, so the absolute volume of packet-congestion will accumulate at twice the rate of either flow, $2p_p.u_1 = p_p(u_1+u_2)$.

Appendix B. Congestion Notification Definition: Further Justification

In Section 3 on the definition of congestion notification, load not capacity was used as the denominator. This also has a subtle significance in the related debate over the design of new transport protocols--typical new protocol designs (e.g. in XCP [I-D.falk-xcp-spec] & Quickstart [RFC4782]) expect the sending transport to communicate its desired flow rate to the network and network elements to progressively subtract from this so that the achievable flow rate emerges at the receiving transport.

Congestion notification with total load in the denominator can serve a similar purpose (though in retrospect not in advance like XCP & QuickStart). Congestion notification is a dimensionless fraction but each source can extract necessary rate information from it because it already knows what its own rate is. Even though congestion notification doesn't communicate a rate explicitly, from each source's point of view congestion notification represents the fraction of the rate it was sending a round trip ago that couldn't (or wouldn't) be served by available resources. After they were sent, all these fractions of each source's offered load added up to the aggregate fraction of offered load seen by the congested resource. So, the source can also know the total excess rate by multiplying total load by congestion level. Therefore congestion notification, as one scale-free dimensionless fraction, implicitly communicates the instantaneous excess flow rate, albeit a RTT ago.

Appendix C. Byte-mode Drop Complicates Policing Congestion Response

This appendix 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 not byte-mode drop.

To be able to police a transport's response to congestion when fairness can only be judged over time and over all an individual's flows, the policer has to have an integrated view of all the congestion an individual (not just one flow) has caused due to all traffic entering the Internet from that individual. This is termed congestion accountability.

But with byte-mode drop, one dropped or marked packet is not necessarily equivalent to another unless you know the MTU that caused it to be dropped/marked. To have an integrated view of a user, we believe congestion policing has to be located at an individual's attachment point to the Internet [Re-TCP]. But from there it cannot know the MTU of each remote queue that caused each drop/mark. Therefore it cannot take an integrated approach to policing all the responses to congestion of all the transports of one individual. Therefore it cannot police anything.

The security/incentive argument _for_ packet-mode drop is similar. Firstly, confining RED to packet-mode drop would not preclude bottleneck policing approaches such as [pBox] as it seems likely they could work just as well by monitoring the volume of dropped bytes rather than packets. Secondly packet-mode dropping/marking naturally allows the congestion notification of packets to be globally meaningful without relying on MTU information held elsewhere.

Because we recommend that a dropped/marked packet should be taken to mean that all the bytes in the packet are dropped/marked, a policer can remain robust against bits being re-divided into different size packets or across different size flows [Rate_fair_Dis]. Therefore policing would work naturally with just simple packet-mode drop in RED.

In summary, 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.

12. References

12.1. 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.

[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.

[RFC2581] Allman, M., Paxson, V., and W. Stevens, "TCP Congestion Control", RFC 2581, April 1999.

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

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

[RFC3448] Handley, M., Floyd, S., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 3448, January 2003.

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

[RFC5033] Floyd, S. and M. Allman, "Specifying New Congestion Control Algorithms", BCP 133, RFC 5033, August 2007.

12.2. Informative References

[CCvarPktSize] Widmer, J., Boutremans, C., and J-Y. Le Boudec, "Congestion Control for Flows with Variable Packet Size", ACM CCR 34(2) 137--151, 2004, <http://doi.acm.org/10.1145/997150.997162>.

[DupTCP] Wischik, D., "Short messages", Royal Society workshop on networks: modelling and control , September 2007, <http://

www.cs.ucl.ac.uk/staff/ucacdjw/Research/shortmsg.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.statslab.cam.ac.uk/~frank/evol.html>.

[I-D.falk-xcp-spec]
Falk, A., "Specification for the Explicit Control Protocol (XCP)", draft-falk-xcp-spec-03 (work in progress), July 2007.

[I-D.floyd-tcpm-ackcc]
Floyd, S. and I. Property, "Adding Acknowledgement Congestion Control to TCP", draft-floyd-tcpm-ackcc-02 (work in progress), November 2007.

[I-D.ietf-pcn-architecture]
Eardley, P., "Pre-Congestion Notification Architecture", draft-ietf-pcn-architecture-03 (work in progress), February 2008.

[I-D.ietf-tcpm-ecnsyn]
Floyd, S., "Adding Explicit Congestion Notification (ECN) Capability to TCP's SYN/ACK Packets", draft-ietf-tcpm-ecnsyn-05 (work in progress), February 2008.

[I-D.ietf-tcpm-rfc2581bis]
Allman, M., "TCP Congestion Control", draft-ietf-tcpm-rfc2581bis-03 (work in progress), September 2007.

[I-D.irtf-iccrg-welzl-congestion-control-open-research]
Papadimitriou, D., "Open Research Issues in Internet Congestion Control", draft-irtf-iccrg-welzl-congestion-control-open-research-00 (work in progress), July 2007.

[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.

[MulTCP] Crowcroft, J. and Ph. Oechslin, "Differentiated End to End Internet Services using a Weighted Proportional Fair Sharing TCP", CCR 28(3) 53--69, July 1998, <http://www.cs.ucl.ac.uk/staff/J.Crowcroft/hipparch/pricing.html>.

[PCNcharter]
IETF, "Congestion and Pre-Congestion Notification (pcn)", IETF w-g charter , Feb 2007, <http://www.ietf.org/html.charters/pcn-charter.html>.

[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://www.icir.org/floyd/papers/red/red.html>.

[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, "RED behavior with different packet sizes", Proc. 5th IEEE Symposium on Computers and Communications (ISCC) 793--799, July 2000, <http://www.icir.org/floyd/red/Elloumi99.pdf>.

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

[RFC4782] Floyd, S., Allman, M., Jain, A., and P. Sarolahti, "Quick-Start for TCP and IP", RFC 4782, January 2007.

[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>.

[Re-TCP] Briscoe, B., Jacquet, A., Moncaster, T., and A. Smith, "Re-ECN: Adding Accountability for Causing Congestion to TCP/IP", draft-briscoe-tsvwg-re-ecn-tcp-05 (work in progress), January 2008.

[WindowPropFair]
Siris, V., "Service Differentiation and Performance of Weighted Window-Based Congestion Control and Packet Marking Algorithms in ECN Networks", Computer Communications 26(4) 314--326, 2002, <http://www.ics.forth.gr/netgroup/publications/weighted_window_control.html>.

[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 in the Internet", IEEE/ACM Transactions on Networking 7(4) 458--472, August 1999, <http://www.aciri.org/floyd/end2end-paper.html>.

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

Author's Address

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

Phone: +44 1473 645196
Email: bob.briscoe@bt.com
URI: http://www.cs.ucl.ac.uk/staff/B.Briscoe/

Full Copyright Statement

Intellectual Property

Acknowledgments

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA). This document was produced using xml2rfc v1.32 (of http://xml.resource.org/) from a source in RFC-2629 XML format.