Network Working Group
Request for Comments: 2914
BCP: 41
Category: Best Current Practice
S. Floyd
ACIRI
September 2000

Congestion Control Principles

Status of this Memo

This document specifies an Internet Best Current Practices for the Internet Community, and requests discussion and suggestions for improvements. Distribution of this memo is unlimited.

Copyright Notice

Abstract

The goal of this document is to explain the need for congestion control in the Internet, and to discuss what constitutes correct congestion control. One specific goal is to illustrate the dangers of neglecting to apply proper congestion control. A second goal is to discuss the role of the IETF in standardizing new congestion control protocols.

1. Introduction

This document draws heavily from earlier RFCs, in some cases reproducing entire sections of the text of earlier documents [RFC2309, RFC2357]. We have also borrowed heavily from earlier publications addressing the need for end-to-end congestion control [FF99].

2. Current standards on congestion control

IETF standards concerning end-to-end congestion control focus either on specific protocols (e.g., TCP [RFC2581], reliable multicast protocols [RFC2357]) or on the syntax and semantics of communications between the end nodes and routers about congestion information (e.g., Explicit Congestion Notification [RFC2481]) or desired quality-of-service (diff-serv)). The role of end-to-end congestion control is also discussed in an Informational RFC on "Recommendations on Queue Management and Congestion Avoidance in the Internet" [RFC2309]. RFC 2309 recommends the deployment of active queue management mechanisms in routers, and the continuation of design efforts towards mechanisms in routers to deal with flows that are unresponsive to congestion notification. We freely borrow from RFC 2309 some of their general discussion of end-to-end congestion control.

In contrast to the RFCs discussed above, this document is a more general discussion of the principles of congestion control. One of the keys to the success of the Internet has been the congestion avoidance mechanisms of TCP. While TCP is still the dominant transport protocol in the Internet, it is not ubiquitous, and there are an increasing number of applications that, for one reason or another, choose not to use TCP. Such traffic includes not only multicast traffic, but unicast traffic such as streaming multimedia that does not require reliability; and traffic such as DNS or routing messages that consist of short transfers deemed critical to the operation of the network. Much of this traffic does not use any form of either bandwidth reservations or end-to-end congestion control. The continued use of end-to-end congestion control by best-effort traffic is critical for maintaining the stability of the Internet.

This document also discusses the general role of the IETF in the standardization of new congestion control protocols.

The discussion of congestion control principles for differentiated services or integrated services is not addressed in this document. Some categories of integrated or differentiated services include a guarantee by the network of end-to-end bandwidth, and as such do not require end-to-end congestion control mechanisms.

3. The development of end-to-end congestion control.

3.1. Preventing congestion collapse.

The Internet protocol architecture is based on a connectionless end-to-end packet service using the IP protocol. The advantages of its connectionless design, flexibility and robustness, have been amply demonstrated. However, these advantages are not without cost: careful design is required to provide good service under heavy load. In fact, lack of attention to the dynamics of packet forwarding can result in severe service degradation or "Internet meltdown". This phenomenon was first observed during the early growth phase of the Internet of the mid 1980s [RFC896], and is technically called "congestion collapse".

The original specification of TCP [RFC793] included window-based flow control as a means for the receiver to govern the amount of data sent by the sender. This flow control was used to prevent overflow of the receiver's data buffer space available for that connection. [RFC793] reported that segments could be lost due either to errors or to network congestion, but did not include dynamic adjustment of the flow-control window in response to congestion.

The original fix for Internet meltdown was provided by Van Jacobson. Beginning in 1986, Jacobson developed the congestion avoidance mechanisms that are now required in TCP implementations [Jacobson88, RFC 2581]. These mechanisms operate in the hosts to cause TCP connections to "back off" during congestion. We say that TCP flows are "responsive" to congestion signals (i.e., dropped packets) from the network. It is these TCP congestion avoidance algorithms that prevent the congestion collapse of today's Internet.

However, that is not the end of the story. Considerable research has been done on Internet dynamics since 1988, and the Internet has grown. It has become clear that the TCP congestion avoidance mechanisms [RFC2581], while necessary and powerful, are not sufficient to provide good service in all circumstances. In addition to the development of new congestion control mechanisms [RFC2357], router-based mechanisms are in development that complement the endpoint congestion avoidance mechanisms.

A major issue that still needs to be addressed is the potential for future congestion collapse of the Internet due to flows that do not use responsible end-to-end congestion control. RFC 896 [RFC896] suggested in 1984 that gateways should detect and 'squelch' misbehaving hosts: "Failure to respond to an ICMP Source Quench message, though, should be regarded as grounds for action by a gateway to disconnect a host. Detecting such failure is non-trivial but is a worthwhile area for further research." Current papers still propose that routers detect and penalize flows that are not employing acceptable end-to-end congestion control [FF99].

3.2. Fairness

In addition to a concern about congestion collapse, there is a concern about 'fairness' for best-effort traffic. Because TCP "backs off" during congestion, a large number of TCP connections can share a single, congested link in such a way that bandwidth is shared reasonably equitably among similarly situated flows. The equitable sharing of bandwidth among flows depends on the fact that all flows are running compatible congestion control algorithms. For TCP, this means congestion control algorithms conformant with the current TCP specification [RFC793, RFC1122, RFC2581].

The issue of fairness among competing flows has become increasingly important for several reasons. First, using window scaling [RFC1323], individual TCPs can use high bandwidth even over highpropagation-delay paths. Second, with the growth of the web, Internet users increasingly want high-bandwidth and low-delay communications, rather than the leisurely transfer of a long file in the background. The growth of best-effort traffic that does not use TCP underscores this concern about fairness between competing best-effort traffic in times of congestion.

The popularity of the Internet has caused a proliferation in the number of TCP implementations. Some of these may fail to implement the TCP congestion avoidance mechanisms correctly because of poor implementation [RFC2525]. Others may deliberately be implemented with congestion avoidance algorithms that are more aggressive in their use of bandwidth than other TCP implementations; this would allow a vendor to claim to have a "faster TCP". The logical consequence of such implementations would be a spiral of increasingly aggressive TCP implementations, or increasingly aggressive transport protocols, leading back to the point where there is effectively no congestion avoidance and the Internet is chronically congested.

There is a well-known way to achieve more aggressive performance without even changing the transport protocol, by changing the level of granularity: open multiple connections to the same place, as has been done in the past by some Web browsers. Thus, instead of a spiral of increasingly aggressive transport protocols, we would instead have a spiral of increasingly aggressive web browsers, or increasingly aggressive applications.

This raises the issue of the appropriate granularity of a "flow", where we define a 'flow' as the level of granularity appropriate for the application of both fairness and congestion control. From RFC 2309: "There are a few 'natural' answers: 1) a TCP or UDP connection (source address/port, destination address/port); 2) a source/destination host pair; 3) a given source host or a given destination host. We would guess that the source/destination host pair gives the most appropriate granularity in many circumstances. The granularity of flows for congestion management is, at least in part, a policy question that needs to be addressed in the wider IETF community."

Again borrowing from RFC 2309, we use the term "TCP-compatible" for a flow that behaves under congestion like a flow produced by a conformant TCP. A TCP-compatible flow is responsive to congestion notification, and in steady-state uses no more bandwidth than a conformant TCP running under comparable conditions (drop rate, RTT, MTU, etc.)

It is convenient to divide flows into three classes: (1) TCP-compatible flows, (2) unresponsive flows, i.e., flows that do not slow down when congestion occurs, and (3) flows that are responsive but are not TCP-compatible. The last two classes contain more aggressive flows that pose significant threats to Internet performance, as we discuss below.

In addition to steady-state fairness, the fairness of the initial slow-start is also a concern. One concern is the transient effect on other flows of a flow with an overly-aggressive slow-start procedure. Slow-start performance is particularly important for the many flows that are short-lived, and only have a small amount of data to transfer.

3.3. Optimizing performance regarding throughput, delay, and loss.

In addition to the prevention of congestion collapse and concerns about fairness, a third reason for a flow to use end-to-end congestion control can be to optimize its own performance regarding throughput, delay, and loss. In some circumstances, for example in environments of high statistical multiplexing, the delay and loss rate experienced by a flow are largely independent of its own sending rate. However, in environments with lower levels of statistical multiplexing or with per-flow scheduling, the delay and loss rate experienced by a flow is in part a function of the flow's own sending rate. Thus, a flow can use end-to-end congestion control to limit the delay or loss experienced by its own packets. We would note, however, that in an environment like the current best-effort Internet, concerns regarding congestion collapse and fairness with competing flows limit the range of congestion control behaviors available to a flow.

4. The role of the standards process

The standardization of a transport protocol includes not only standardization of aspects of the protocol that could affect interoperability (e.g., information exchanged by the end-nodes), but also standardization of mechanisms deemed critical to performance (e.g., in TCP, reduction of the congestion window in response to a packet drop). At the same time, implementation-specific details and other aspects of the transport protocol that do not affect interoperability and do not significantly interfere with performance do not require standardization. Areas of TCP that do not require standardization include the details of TCP's Fast Recovery procedure after a Fast Retransmit [RFC2582]. The appendix uses examples from TCP to discuss in more detail the role of the standards process in the development of congestion control.

4.1. The development of new transport protocols.

In addition to addressing the danger of congestion collapse, the standardization process for new transport protocols takes care to avoid a congestion control 'arms race' among competing protocols. As an example, in RFC 2357 [RFC2357] the TSV Area Directors and their Directorate outline criteria for the publication as RFCs of Internet-Drafts on reliable multicast transport protocols. From [RFC2357]: "A particular concern for the IETF is the impact of reliable multicast traffic on other traffic in the Internet in times of congestion, in particular the effect of reliable multicast traffic on competing TCP traffic.... The challenge to the IETF is to encourage research and implementations of reliable multicast, and to enable the needs of applications for reliable multicast to be met as expeditiously as possible, while at the same time protecting the Internet from the congestion disaster or collapse that could result from the widespread use of applications with inappropriate reliable multicast mechanisms."

The list of technical criteria that must be addressed by RFCs on new reliable multicast transport protocols include the following: "Is there a congestion control mechanism? How well does it perform? When does it fail? Note that congestion control mechanisms that operate on the network more aggressively than TCP will face a great burden of proof that they don't threaten network stability."

It is reasonable to expect that these concerns about the effect of new transport protocols on competing traffic will apply not only to reliable multicast protocols, but to unreliable unicast, reliable unicast, and unreliable multicast traffic as well.

4.2. Application-level issues that affect congestion control

The specific issue of a browser opening multiple connections to the same destination has been addressed by RFC 2616 [RFC2616], which states in Section 8.1.4 that "Clients that use persistent connections SHOULD limit the number of simultaneous connections that they maintain to a given server. A single-user client SHOULD NOT maintain more than 2 connections with any server or proxy."

4.3. New developments in the standards process

The most obvious developments in the IETF that could affect the evolution of congestion control are the development of integrated and differentiated services [RFC2212, RFC2475] and of Explicit Congestion Notification (ECN) [RFC2481]. However, other less dramatic developments are likely to affect congestion control as well.

One such effort is that to construct Endpoint Congestion Management [BS00], to enable multiple concurrent flows from a sender to the same receiver to share congestion control state. By allowing multiple connections to the same destination to act as one flow in terms of end-to-end congestion control, a Congestion Manager could allow individual connections slow-starting to take advantage of previous information about the congestion state of the end-to-end path. Further, the use of a Congestion Manager could remove the congestion control dangers of multiple flows being opened between the same source/destination pair, and could perhaps be used to allow a browser to open many simultaneous connections to the same destination.

5. A description of congestion collapse

This section discusses congestion collapse from undelivered packets in some detail, and shows how unresponsive flows could contribute to congestion collapse in the Internet. This section draws heavily on material from [FF99].

Informally, congestion collapse occurs when an increase in the network load results in a decrease in the useful work done by the network. As discussed in Section 3, congestion collapse was first reported in the mid 1980s [RFC896], and was largely due to TCP connections unnecessarily retransmitting packets that were either in transit or had already been received at the receiver. We call the congestion collapse that results from the unnecessary retransmission of packets classical congestion collapse. Classical congestion collapse is a stable condition that can result in throughput that is a small fraction of normal [RFC896]. Problems with classical congestion collapse have generally been corrected by the timer improvements and congestion control mechanisms in modern implementations of TCP [Jacobson88].

A second form of potential congestion collapse occurs due to undelivered packets. Congestion collapse from undelivered packets arises when bandwidth is wasted by delivering packets through the network that are dropped before reaching their ultimate destination. This is probably the largest unresolved danger with respect to congestion collapse in the Internet today. Different scenarios can result in different degrees of congestion collapse, in terms of the fraction of the congested links' bandwidth used for productive work. The danger of congestion collapse from undelivered packets is due primarily to the increasing deployment of open-loop applications not using end-to-end congestion control. Even more destructive would be best-effort applications that *increase* their sending rate in response to an increased packet drop rate (e.g., automatically using an increased level of FEC).

Table 1 gives the results from a scenario with congestion collapse from undelivered packets, where scarce bandwidth is wasted by packets that never reach their destination. The simulation uses a scenario with three TCP flows and one UDP flow competing over a congested 1.5 Mbps link. The access links for all nodes are 10 Mbps, except that the access link to the receiver of the UDP flow is 128 Kbps, only 9% of the bandwidth of shared link. When the UDP source rate exceeds 128 Kbps, most of the UDP packets will be dropped at the output port to that final link.

UDP Arrival Rate	UDP Goodput	TCP Goodput	Total Goodput
0.7	0.7	98.5	99.2
1.8	1.7	97.3	99.1
2.6	2.6	96.0	98.6
5.3	5.2	92.7	97.9
8.8	8.4	87.1	95.5
10.5	8.4	84.8	93.2
13.1	8.4	81.4	89.8
17.5	8.4	77.3	85.7
26.3	8.4	64.5	72.8
52.6	8.4	38.1	46.4
58.4	8.4	32.8	41.2
65.7	8.4	28.5	36.8
75.1	8.4	19.7	28.1
87.6	8.4	11.3	19.7
105.2	8.4	3.4	11.8
131.5	8.4	2.4	10.7

Table 1. A simulation with three TCP flows and one UDP flow.

Table 1 shows the UDP arrival rate from the sender, the UDP goodput (defined as the bandwidth delivered to the receiver), the TCP goodput (as delivered to the TCP receivers), and the aggregate goodput on the congested 1.5 Mbps link. Each rate is given as a fraction of the bandwidth of the congested link. As the UDP source rate increases, the TCP goodput decreases roughly linearly, and the UDP goodput is nearly constant. Thus, as the UDP flow increases its offered load, its only effect is to hurt the TCP and aggregate goodput. On the congested link, the UDP flow ultimately 'wastes' the bandwidth that could have been used by the TCP flow, and reduces the goodput in the network as a whole down to a small fraction of the bandwidth of the congested link.

The simulations in Table 1 illustrate both unfairness and congestion collapse. As [FF99] discusses, compatible congestion control is not the only way to provide fairness; per-flow scheduling at the congested routers is an alternative mechanism at the routers that guarantees fairness. However, as discussed in [FF99], per-flow scheduling can not be relied upon to prevent congestion collapse.

There are only two alternatives for eliminating the danger of congestion collapse from undelivered packets. The first alternative for preventing congestion collapse from undelivered packets is the use of effective end-to-end congestion control by the end nodes. More specifically, the requirement would be that a flow avoid a pattern of significant losses at links downstream from the first congested link on the path. (Here, we would consider any link a 'congested link' if any flow is using bandwidth that would otherwise be used by other traffic on the link.) Given that an end-node is generally unable to distinguish between a path with one congested link and a path with multiple congested links, the most reliable way for a flow to avoid a pattern of significant losses at a downstream congested link is for the flow to use end-to-end congestion control, and reduce its sending rate in the presence of loss.

A second alternative for preventing congestion collapse from undelivered packets would be a guarantee by the network that packets accepted at a congested link in the network will be delivered all the way to the receiver [RFC2212, RFC2475]. We note that the choice between the first alternative of end-to-end congestion control and the second alternative of end-to-end bandwidth guarantees does not have to be an either/or decision; congestion collapse can be prevented by the use of effective end-to-end congestion by some of the traffic, and the use of end-to-end bandwidth guarantees from the network for the rest of the traffic.

6. Forms of end-to-end congestion control

This document has discussed concerns about congestion collapse and about fairness with TCP for new forms of congestion control. This does not mean, however, that concerns about congestion collapse and fairness with TCP necessitate that all best-effort traffic deploy congestion control based on TCP's Additive-Increase Multiplicative-Decrease (AIMD) algorithm of reducing the sending rate in half in response to each packet drop. This section separately discusses the implications of these two concerns of congestion collapse and fairness with TCP.

6.1. End-to-end congestion control for avoiding congestion collapse.

The avoidance of congestion collapse from undelivered packets requires that flows avoid a scenario of a high sending rate, multiple congested links, and a persistent high packet drop rate at the downstream link. Because congestion collapse from undelivered packets consists of packets that waste valuable bandwidth only to be dropped downstream, this form of congestion collapse is not possible in an environment where each flow traverses only one congested link, or where only a small number of packets are dropped at links downstream of the first congested link. Thus, any form of congestion control that successfully avoids a high sending rate in the presence of a high packet drop rate should be sufficient to avoid congestion collapse from undelivered packets.

We would note that the addition of Explicit Congestion Notification (ECN) to the IP architecture would not, in and of itself, remove the danger of congestion collapse for best-effort traffic. ECN allows routers to set a bit in packet headers as an indication of congestion to the end-nodes, rather than being forced to rely on packet drops to indicate congestion. However, with ECN, packet-marking would replace packet-dropping only in times of moderate congestion. In particular, when congestion is heavy, and a router's buffers overflow, the router has no choice but to drop arriving packets.

6.2. End-to-end congestion control for fairness with TCP.

The concern expressed in [RFC2357] about fairness with TCP places a significant though not crippling constraint on the range of viable end-to-end congestion control mechanisms for best-effort traffic. An environment with per-flow scheduling at all congested links would isolate flows from each other, and eliminate the need for congestion control mechanisms to be TCP-compatible. An environment with differentiated services, where flows marked as belonging to a certain diff-serv class would be scheduled in isolation from best-effort traffic, could allow the emergence of an entire diff-serv class of traffic where congestion control was not required to be TCP-compatible. Similarly, a pricing-controlled environment, or a diff-serv class with its own pricing paradigm, could supercede the concern about fairness with TCP. However, for the current Internet environment, where other best-effort traffic could compete in a FIFO queue with TCP traffic, the absence of fairness with TCP could lead to one flow 'starving out' another flow in a time of high congestion, as was illustrated in Table 1 above.

However, the list of TCP-compatible congestion control procedures is not limited to AIMD with the same increase/ decrease parameters as TCP. Other TCP-compatible congestion control procedures include rate-based variants of AIMD; AIMD with different sets of increase/decrease parameters that give the same steady-state behavior; equation-based congestion control where the sender adjusts its sending rate in response to information about the long-term packet drop rate; layered multicast where receivers subscribe and unsubscribe from layered multicast groups; and possibly other forms that we have not yet begun to consider.

7. Acknowledgements

Much of this document draws directly on previous RFCs addressing end-to-end congestion control. This attempts to be a summary of ideas that have been discussed for many years, and by many people. In particular, acknowledgement is due to the members of the End-to-End Research Group, the Reliable Multicast Research Group, and the Transport Area Directorate. This document has also benefited from discussion and feedback from the Transport Area Working Group. Particular thanks are due to Mark Allman for feedback on an earlier version of this document.

8. References

[BS00] Balakrishnan H. and S. Seshan, "The Congestion Manager", Work in Progress.

[DMKM00] Dawkins, S., Montenegro, G., Kojo, M. and V. Magret, "End-to-end Performance Implications of Slow Links", Work in Progress.

[FF99] Floyd, S. and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet", IEEE/ACM Transactions on Networking, August 1999. URL http://www.aciri.org/floyd/end2end-paper.html

[HPF00] Handley, M., Padhye, J. and S. Floyd, "TCP Congestion Window Validation", RFC 2861, June 2000.

[Jacobson88] V. Jacobson, Congestion Avoidance and Control, ACM SIGCOMM '88, August 1988.

[RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC896] Nagle, J., "Congestion Control in IP/TCP", RFC 896, January 1984.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts -- Communication Layers", STD 3, RFC 1122, October 1989.

[RFC1323] Jacobson, V., Braden, R. and D. Borman, "TCP Extensions for High Performance", RFC 1323, May 1992.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2212] Shenker, S., Partridge, C. and R. Guerin, "Specification of Guaranteed Quality of Service", RFC 2212, September 1997.

[RFC2309] Braden, R., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K.K., Shenker, S., Wroclawski, J., and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, April 1998.

[RFC2357] Mankin, A., Romanow, A., Bradner, S. and V. Paxson, "IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols", RFC 2357, June 1998.

[RFC2414] Allman, M., Floyd, S. and C. Partridge, "Increasing TCP's Initial Window", RFC 2414, September 1998.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z. and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

[RFC2481] Ramakrishnan K. and S. Floyd, "A Proposal to add Explicit Congestion Notification (ECN) to IP", RFC 2481, January 1999.

[RFC2525] Paxson, V., Allman, M., Dawson, S., Fenner, W., Griner, J., Heavens, I., Lahey, K., Semke, J. and B. Volz, "Known TCP Implementation Problems", RFC 2525, March 1999.

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

[RFC2582] Floyd, S. and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm", RFC 2582, April 1999.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[SCWA99] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson, TCP Congestion Control with a Misbehaving Receiver, ACM Computer Communications Review, October 1999.

[TCPB98] Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Seshan, Mark Stemm, and Randy H. Katz, TCP Behavior of a Busy Internet Server: Analysis and Improvements, IEEE Infocom, March 1998. Available from: "http://www.cs.berkeley.edu/~hari/papers/infocom98.ps.gz".

[TCPF98] Dong Lin and H.T. Kung, TCP Fast Recovery Strategies: Analysis and Improvements, IEEE Infocom, March 1998. Available from: "http://www.eecs.harvard.edu/networking/papers/infocom-tcp-final-198.pdf".

9. TCP-Specific issues

In this section we discuss some of the particulars of TCP congestion control, to illustrate a realization of the congestion control principles, including some of the details that arise when incorporating them into a production transport protocol.

9.1. Slow-start.

The TCP sender can not open a new connection by sending a large burst of data (e.g., a receiver's advertised window) all at once. The TCP sender is limited by a small initial value for the congestion window. During slow-start, the TCP sender can increase its sending rate by at most a factor of two in one roundtrip time. Slow-start ends when congestion is detected, or when the sender's congestion window is greater than the slow-start threshold ssthresh.

An issue that potentially affects global congestion control, and therefore has been explicitly addressed in the standards process, includes an increase in the value of the initial window [RFC2414,RFC2581].

Issues that have not been addressed in the standards process, and are generally considered not to require standardization, include such issues as the use (or non-use) of rate-based pacing, and mechanisms for ending slow-start early, before the congestion window reaches ssthresh. Such mechanisms result in slow-start behavior that is as conservative or more conservative than standard TCP.

9.2. Additive Increase, Multiplicative Decrease.

In the absence of congestion, the TCP sender increases its congestion window by at most one packet per roundtrip time. In response to a congestion indication, the TCP sender decreases its congestion window by half. (More precisely, the new congestion window is half of the minimum of the congestion window and the receiver's advertised window.)

An issue that potentially affects global congestion control, and therefore would be likely to be explicitly addressed in the standards process, would include a proposed addition of congestion control for the return stream of 'pure acks'.

An issue that has not been addressed in the standards process, and is generally not considered to require standardization, would be a change to the congestion window to apply as an upper bound on the number of bytes presumed to be in the pipe, instead of applying as a sliding window starting from the cumulative acknowledgement. (Clearly, the receiver's advertised window applies as a sliding window starting from the cumulative acknowledgement field, because packets received above the cumulative acknowledgement field are held in TCP's receive buffer, and have not been delivered to the application. However, the congestion window applies to the number of packets outstanding in the pipe, and does not necessarily have to include packets that have been received out-of-order by the TCP receiver.)

9.3. Retransmit timers.

The TCP sender sets a retransmit timer to infer that a packet has been dropped in the network. When the retransmit timer expires, the sender infers that a packet has been lost, sets ssthresh to half of the current window, and goes into slow-start, retransmitting the lost packet. If the retransmit timer expires because no acknowledgement has been received for a retransmitted packet, the retransmit timer is also "backed-off", doubling the value of the next retransmit timeout interval.

An issue that potentially affects global congestion control, and therefore would be likely to be explicitly addressed in the standards process, might include a modified mechanism for setting the retransmit timer that could significantly increase the number of retransmit timers that expire prematurely, when the acknowledgement has not yet arrived at the sender, but in fact no packets have been dropped. This could be of concern to the Internet standards process because retransmit timers that expire prematurely could lead to an increase in the number of packets unnecessarily transmitted on a congested link.

9.4. Fast Retransmit and Fast Recovery.

After seeing three duplicate acknowledgements, the TCP sender infers a packet loss. The TCP sender sets ssthresh to half of the current window, reduces the congestion window to at most half of the previous window, and retransmits the lost packet.

An issue that potentially affects global congestion control, and therefore would be likely to be explicitly addressed in the standards process, might include a proposal (if there was one) for inferring a lost packet after only one or two duplicate acknowledgements. If poorly designed, such a proposal could lead to an increase in the number of packets unnecessarily transmitted on a congested path.

An issue that has not been addressed in the standards process, and would not be expected to require standardization, would be a proposal to send a "new" or presumed-lost packet in response to a duplicate or partial acknowledgement, if allowed by the congestion window. An example of this would be sending a new packet in response to a single duplicate acknowledgement, to keep the ‘ack clock’ going in case no further acknowledgements would have arrived. Such a proposal is an example of a beneficial change that does not involve interoperability and does not affect global congestion control, and that therefore could be implemented by vendors without requiring the intervention of the IETF standards process. (This issue has in fact been addressed in [DMKM00], which suggests that "researchers may wish to experiment with injecting new traffic into the network when duplicate acknowledgements are being received, as described in [TCPB98] and [TCPF98]."

9.5. Other aspects of TCP congestion control.

Other aspects of TCP congestion control that have not been discussed in any of the sections above include TCP’s recovery from an idle or application-limited period [HPF00].

10. Security Considerations

This document has been about the risks associated with congestion control, or with the absence of congestion control. Section 3.2 discusses the potentials for unfairness if competing flows don’t use compatible congestion control mechanisms, and Section 5 considers the dangers of congestion collapse if flows don’t use end-to-end congestion control.

Because this document does not propose any specific congestion control mechanisms, it is also not necessary to present specific security measures associated with congestion control. However, we would note that there are a range of security considerations associated with congestion control that should be considered in IETF documents.

For example, individual congestion control mechanisms should be as robust as possible to the attempts of individual end-nodes to subvert end-to-end congestion control [SCWA99]. This is a particular concern in multicast congestion control, because of the far-reaching distribution of the traffic and the greater opportunities for individual receivers to fail to report congestion.

RFC 2309 also discussed the potential dangers to the Internet of unresponsive flows, that is, flows that don't reduce their sending rate in the presence of congestion, and describes the need for mechanisms in the network to deal with flows that are unresponsive to congestion notification. We would note that there is still a need for research, engineering, measurement, and deployment in these areas.

Because the Internet aggregates very large numbers of flows, the risk to the whole infrastructure of subverting the congestion control of a few individual flows is limited. Rather, the risk to the infrastructure would come from the widespread deployment of many end-nodes subverting end-to-end congestion control.

AUTHOR'S ADDRESS

Sally Floyd
AT&T Center for Internet Research at ICSI (ACIRI)

Phone: +1 (510) 642-4274 x189
EMail: floyd@aciri.org
URL: http://www.aciri.org/floyd/

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.