Network Working Group
Request for Comments: 3155
BCP: 50
Category: Best Current Practice

S. Dawkins
G. Montenegro
M. Kojo
V. Magret
N. Vaidya

August 2001

# End-to-end Performance Implications of Links with Errors

## Status of this Memo



## Copyright Notice



## Abstract

This document discusses the specific TCP mechanisms that are problematic in environments with high uncorrected error rates, and discusses what can be done to mitigate the problems without introducing intermediate devices into the connection.

## Table of Contents

## 1.0 Introduction

The rapidly-growing Internet is being accessed by an increasingly wide range of devices over an increasingly wide variety of links. At least some of these links do not provide the degree of reliability that hosts expect, and this expansion into unreliable links causes some Internet protocols, especially TCP [RFC793], to perform poorly.

Specifically, TCP congestion control [RFC2581], while appropriate for connections that lose traffic primarily because of congestion and buffer exhaustion, interacts badly with uncorrected errors when TCP connections traverse links with high uncorrected error rates. The result is that sending TCPs may spend an excessive amount of time waiting for acknowledgement that do not arrive, and then, although these losses are not due to congestion-related buffer exhaustion, the sending TCP transmits at substantially reduced traffic levels as it probes the network to determine "safe" traffic levels.

This document does not address issues with other transport protocols, for example, UDP.

Congestion avoidance in the Internet is based on an assumption that most packet losses are due to congestion. TCP's congestion avoidance strategy treats the absence of acknowledgement as a congestion signal. This has worked well since it was introduced in 1988 [VJ-DCAC], because most links and subnets have relatively low error rates in normal operation, and congestion is the primary cause of loss in these environments. However, links and subnets that do not enjoy low uncorrected error rates are becoming more prevalent in parts of the Internet. In particular, these include terrestrial and satellite wireless links. Users relying on traffic traversing these links may see poor performance because their TCP connections are spending excessive time in congestion avoidance and/or slow start procedures triggered by packet losses due to transmission errors.

The recommendations in this document aim at improving utilization of available path capacity over such high error-rate links in ways that do not threaten the stability of the Internet.

Applications use TCP in very different ways, and these have interactions with TCP's behavior [RFC2861]. Nevertheless, it is possible to make some basic assumptions about TCP flows. Accordingly, the mechanisms discussed here are applicable to all uses of TCP, albeit in varying degrees according to different scenarios (as noted where appropriate).

This recommendation is based on the explicit assumption that major changes to the entire installed base of routers and hosts are not a practical possibility. This constrains any changes to hosts that are directly affected by errored links.

## 1.1 Should you be reading this recommendation?

All known subnetwork technologies provide an "imperfect" subnetwork service - the bit error rate is non-zero. But there's no obvious way for end stations to tell the difference between packets discarded due to congestion and losses due to transmission errors.

If a directly-attached subnetwork is reporting transmission errors to a host, these reports matter, but we can't rely on explicit transmission error reports to both hosts.

Another way of deciding if a subnetwork should be considered to have a "high error rate" is by appealing to mathematics.

An approximate formula for the TCP Reno response function is given in [PFTK98]:

$$T = \frac{s}{RTT\sqrt{2p/3} + tRTO(3\sqrt{3p/8})p(1 + 32p^2)}$$

where

T = the sending rate in bytes per second
s = the packet size in bytes
RTT = round-trip time in seconds
tRTO = TCP retransmit timeout value in seconds
p = steady-state packet loss rate

If one plugs in an observed packet loss rate, does the math and then sees predicted bandwidth utilization that is greater than the link speed, the connection will not benefit from recommendations in this document, because the level of packet losses being encountered won't affect the ability of TCP to utilize the link. If, however, the predicted bandwidth is less than the link speed, packet losses are affecting the ability of TCP to utilize the link.

If further investigation reveals a subnetwork with significant transmission error rates, the recommendations in this document will improve the ability of TCP to utilize the link.

A few caveats are in order, when doing this calculation:

(1) the RTT is the end-to-end RTT, not the link RTT.
(2) Max(1.0, 4*RTT) can be substituted as a simplification for tRTO.
(3) losses may be bursty - a loss rate measured over an interval that includes multiple bursty loss events may understate the impact of these loss events on the sending rate.

## 1.2 Relationship of this recommendation to PEPs

This document discusses end-to-end mechanisms that do not require TCP-level awareness by intermediate nodes. This places severe limitations on what the end nodes can know about the nature of losses that are occurring between the end nodes. Attempts to apply heuristics to distinguish between congestion and transmission error have not been successful [BV97, BV98, BV98a]. This restriction is relaxed in an informational document on Performance Enhancing Proxies (PEPs) [RFC3135]. Because PEPs can be placed on boundaries where network characteristics change dramatically, PEPs have an additional opportunity to improve performance over links with uncorrected errors.

However, generalized use of PEPs contravenes the end-to-end principle and is highly undesirable given their deleterious implications, which include the following: lack of fate sharing (a PEP adds a third point of failure besides the endpoints themselves), end-to-end reliability and diagnostics, preventing end-to-end security (particularly network layer security such as IPsec), mobility (handoffs are much more complex because state must be transferred), asymmetric routing (PEPs typically require being on both the forward and reverse paths of a connection), scalability (PEPs add more state to maintain), QoS transparency and guarantees.

Not every type of PEP has all the drawbacks listed above. Nevertheless, the use of PEPs may have very serious consequences which must be weighed carefully.

## 1.3 Relationship of this recommendation to Link Layer Mechanisms

This recommendation is for use with TCP over subnetwork technologies (link layers) that have already been deployed. Subnetworks that are intended to carry Internet protocols, but have not been completely specified are the subject of a best common practices (BCP) document which has been developed or is under development by the Performance

Implications of Link Characteristics WG (PILC) [PILC-WEB]. This last document is aimed at designers who still have the opportunity to reduce the number of uncorrected errors TCP will encounter.

## 2.0 Errors and Interactions with TCP Mechanisms

A TCP sender adapts its use of network path capacity based on feedback from the TCP receiver. As TCP is not able to distinguish between losses due to congestion and losses due to uncorrected errors, it is not able to accurately determine available path capacity in the presence of significant uncorrected errors.

### 2.1 Slow Start and Congestion Avoidance [RFC2581]

Slow Start and Congestion Avoidance [RFC2581] are essential to the current stability of the Internet. These mechanisms were designed to accommodate networks that do not provide explicit congestion notification. Although experimental mechanisms such as [RFC2481] are moving in the direction of explicit congestion notification, the effect of ECN on ECN-aware TCPs is essentially the same as the effect of implicit congestion notification through congestion-related loss, except that ECN provides this notification before packets are lost, and must then be retransmitted.

TCP connections experiencing high error rates on their paths interact badly with Slow Start and with Congestion Avoidance, because high error rates make the interpretation of losses ambiguous - the sender cannot know whether detected losses are due to congestion or to data corruption. TCP makes the "safe" choice and assumes that the losses are due to congestion.

- Whenever sending TCPs receive three out-of-order acknowledgement, they assume the network is mildly congested and invoke fast retransmit/fast recovery (described below).

- Whenever TCP's retransmission timer expires, the sender assumes that the network is congested and invokes slow start.

- Less-reliable link layers often use small link MTUs. This slows the rate of increase in the sender's window size during slow start, because the sender's window is increased in units of segments. Small link MTUs alone don't improve reliability. Path MTU discovery [RFC1191] must also be used to prevent fragmentation. Path MTU discovery allows the most rapid opening of the sender's window size during slow start, but a number of round trips may still be required to open the window completely.

Recommendation: Any standards-conformant TCP will implement Slow Start and Congestion Avoidance, which are MUSTs in STD 3 [RFC1122]. Recommendations in this document will not interfere with these mechanisms.

## 2.2 Fast Retransmit and Fast Recovery [RFC2581]

TCP provides reliable delivery of data as a byte-stream to an application, so that when a segment is lost (whether due to either congestion or transmission loss), the receiver TCP implementation must wait to deliver data to the receiving application until the missing data is received. The receiver TCP implementation detects missing segments by segments arriving with out-of-order sequence numbers.

TCPs should immediately send an acknowledgement when data is received out-of-order [RFC2581], providing the next expected sequence number with no delay, so that the sender can retransmit the required data as quickly as possible and the receiver can resume delivery of data to the receiving application. When an acknowledgement carries the same expected sequence number as an acknowledgement that has already been sent for the last in-order segment received, these acknowledgement are called "duplicate ACKs".

Because IP networks are allowed to reorder packets, the receiver may send duplicate acknowledgments for segments that arrive out of order due to routing changes, link-level retransmission, etc. When a TCP sender receives three duplicate ACKs, fast retransmit [RFC2581] allows it to infer that a segment was lost. The sender retransmits what it considers to be this lost segment without waiting for the full retransmission timeout, thus saving time.

After a fast retransmit, a sender halves its congestion window and invokes the fast recovery [RFC2581] algorithm, whereby it invokes congestion avoidance from a halved congestion window, but does not invoke slow start from a one-segment congestion window as it would do after a retransmission timeout. As the sender is still receiving dupacks, it knows the receiver is receiving packets sent, so the full reduction after a timeout when no communication has been received is not called for. This relatively safe optimization also saves time.

It is important to be realistic about the maximum throughput that TCP can have over a connection that traverses a high error-rate link. In general, TCP will increase its congestion window beyond the delay-bandwidth product. TCP's congestion avoidance strategy is additive-increase, multiplicative-decrease, which means that if additional errors are encountered before the congestion window recovers completely from a 50-percent reduction, the effect can be a "downward

spiral" of the congestion window due to additional 50-percent reductions. Even using Fast Retransmit/Fast Recovery, the sender will halve the congestion window each time a window contains one or more segments that are lost, and will re-open the window by one additional segment for each congestion window's worth of acknowledgement received.

If a connection's path traverses a link that loses one or more segments during this recovery period, the one-half reduction takes place again, this time on a reduced congestion window - and this downward spiral will continue to hold the congestion window below path capacity until the connection is able to recover completely by additive increase without experiencing loss.

Of course, no downward spiral occurs if the error rate is constantly high and the congestion window always remains small; the multiplicative-increase "slow start" will be exited early, and the congestion window remains low for the duration of the TCP connection. In links with high error rates, the TCP window may remain rather small for long periods of time.

Not all causes of small windows are related to errors. For example, HTTP/1.0 commonly closes TCP connections to indicate boundaries between requested resources. This means that these applications are constantly closing "trained" TCP connections and opening "untrained" TCP connections which will execute slow start, beginning with one or two segments. This can happen even with HTTP/1.1, if webmasters configure their HTTP/1.1 servers to close connections instead of waiting to see if the connection will be useful again.

A small window - especially a window of less than four segments - effectively prevents the sender from taking advantage of Fast Retransmits. Moreover, efficient recovery from multiple losses within a single window requires adoption of new proposals (NewReno [RFC2582]).

Recommendation: Implement Fast Retransmit and Fast Recovery at this time. This is a widely-implemented optimization and is currently at Proposed Standard level. [RFC2488] recommends implementation of Fast Retransmit/Fast Recovery in satellite environments.

### 2.3 Selective Acknowledgements [RFC2018, RFC2883]

Selective Acknowledgements [RFC2018] allow the repair of multiple segment losses per window without requiring one (or more) round-trips per loss.

[RFC2883] proposes a minor extension to SACK that allows receiving TCPs to provide more information about the order of delivery of segments, allowing "more robust operation in an environment of reordered packets, ACK loss, packet replication, and/or early retransmit timeouts". Unless explicitly stated otherwise, in this document, "Selective Acknowledgements" (or "SACK") refers to the combination of [RFC2018] and [RFC2883].

Selective acknowledgments are most useful in LFNs ("Long Fat Networks") because of the long round trip times that may be encountered in these environments, according to Section 1.1 of [RFC1323], and are especially useful if large windows are required, because there is a higher probability of multiple segment losses per window.

On the other hand, if error rates are generally low but occasionally higher due to channel conditions, TCP will have the opportunity to increase its window to larger values during periods of improved channel conditions between bursts of errors. When bursts of errors occur, multiple losses within a window are likely to occur. In this case, SACK would provide benefits in speeding the recovery and preventing unnecessary reduction of the window size.

Recommendation: Implement SACK as specified in [RFC2018] and updated by [RFC2883], both Proposed Standards. In cases where SACK cannot be enabled for both sides of a connection, TCP senders may use NewReno [RFC2582] to better handle partial ACKs and multiple losses within a single window.

## 3.0 Summary of Recommendations

The Internet does not provide a widely-available loss feedback mechanism that allows TCP to distinguish between congestion loss and transmission error. Because congestion affects all traffic on a path while transmission loss affects only the specific traffic encountering uncorrected errors, avoiding congestion has to take precedence over quickly repairing transmission errors. This means that the best that can be achieved without new feedback mechanisms is minimizing the amount of time that is spent unnecessarily in congestion avoidance.

The Fast Retransmit/Fast Recovery mechanism allows quick repair of loss without giving up the safety of congestion avoidance. In order for Fast Retransmit/Fast Recovery to work, the window size must be large enough to force the receiver to send three duplicate acknowledgments before the retransmission timeout interval expires, forcing full TCP slow-start.

Selective Acknowledgements (SACK) extend the benefit of Fast Retransmit/Fast Recovery to situations where multiple segment losses in the window need to be repaired more quickly than can be accomplished by executing Fast Retransmit for each segment loss, only to discover the next segment loss.

These mechanisms are not limited to wireless environments. They are usable in all environments.

## 4.0 Topics For Further Work

"Limited Transmit" [RFC3042] has been specified as an optimization extending Fast Retransmit/Fast Recovery for TCP connections with small congestion windows that will not trigger three duplicate acknowledgments. This specification is deemed safe, and it also provides benefits for TCP connections that experience a large amount of packet (data or ACK) loss. Implementors should evaluate this standards track specification for TCP in loss environments.

Delayed Duplicate Acknowledgements [MV97, VMPM99] attempts to prevent TCP-level retransmission when link-level retransmission is still in progress, adding additional traffic to the network. This proposal is worthy of additional study, but is not recommended at this time, because we don't know how to calculate appropriate amounts of delay for an arbitrary network topology.

It is not possible to use explicit congestion notification [RFC2481] as a surrogate for explicit transmission error notification (no matter how much we wish it was!). Some mechanism to provide explicit notification of transmission error would be very helpful. This might be more easily provided in a PEP environment, especially when the PEP is the "first hop" in a connection path, because current checksum mechanisms do not distinguish between transmission error to a payload and transmission error to the header. Furthermore, if the header is damaged, sending explicit transmission error notification to the right endpoint is problematic.

Losses that take place on the ACK stream, especially while a TCP is learning network characteristics, can make the data stream quite bursty (resulting in losses on the data stream, as well). Several ways of limiting this burstiness have been proposed, including TCP transmit pacing at the sender and ACK rate control within the network.

"Appropriate Byte Counting" (ABC) [ALL99], has been proposed as a way of opening the congestion window based on the number of bytes that have been successfully transfered to the receiver, giving more appropriate behavior for application protocols that initiate

connections with relatively short packets. For SMTP [RFC2821], for instance, the client might send a short HELO packet, a short MAIL packet, one or more short RCPT packets, and a short DATA packet - followed by the entire mail body sent as maximum-length packets. An ABC TCP sender would not use ACKs for each of these short packets to increase the congestion window to allow additional full-length packets. ABC is worthy of additional study, but is not recommended at this time, because ABC can lead to increased burstiness when acknowledgments are lost.

### 4.1 Achieving, and maintaining, large windows

The recommendations described in this document will aid TCPs in injecting packets into ERROREd connections as fast as possible without destabilizing the Internet, and so optimizing the use of available bandwidth.

In addition to these TCP-level recommendations, there is still additional work to do at the application level, especially with the dominant application protocol on the World Wide Web, HTTP.

HTTP/1.0 (and earlier versions) closes TCP connections to signal a receiver that all of a requested resource had been transmitted. Because WWW objects tend to be small in size [MOGUL], TCPs carrying HTTP/1.0 traffic experience difficulty in "training" on available path capacity (a substantial portion of the transfer has already happened by the time TCP exits slow start).

Several HTTP modifications have been introduced to improve this interaction with TCP ("persistent connections" in HTTP/1.0, with improvements in HTTP/1.1 [RFC2616]). For a variety of reasons, many HTTP interactions are still HTTP/1.0-style - relatively short-lived.

Proposals which reuse TCP congestion information across connections, like TCP Control Block Interdependence [RFC2140], or the more recent Congestion Manager [BS00] proposal, will have the effect of making multiple parallel connections impact the network as if they were a single connection, "trained" after a single startup transient. These proposals are critical to the long-term stability of the Internet, because today's users always have the choice of clicking on the "reload" button in their browsers and cutting off TCP's exponential backoff - replacing connections which are building knowledge of the available bandwidth with connections with no knowledge at all.

## 5.0 Security Considerations

A potential vulnerability introduced by Fast Retransmit/Fast Recovery is (as pointed out in [RFC2581]) that an attacker may force TCP connections to grind to a halt, or, more dangerously, behave more aggressively. The latter possibility may lead to congestion collapse, at least in some regions of the network.

Selective acknowledgments is believed to neither strengthen nor weaken TCP's current security properties [RFC2018].

Given that the recommendations in this document are performed on an end-to-end basis, they continue working even in the presence of end-to-end IPsec. This is in direct contrast with mechanisms such as PEP's which are implemented in intermediate nodes (section 1.2).

## 6.0 IANA Considerations

This document is a pointer to other, existing IETF standards. There are no new IANA considerations.

## 7.0 Acknowledgements

This recommendation has grown out of RFC 2757, "Long Thin Networks", which was in turn based on work done in the IETF TCPSAT working group. The authors are indebted to the active members of the PILC working group. In particular, Mark Allman and Lloyd Wood gave us copious and insightful feedback, and Dan Grossman and Jamshid Mahdavi provided text replacements.

## References

[ALL99] M. Allman, "TCP Byte Counting Refinements," ACM Computer Communication Review, Volume 29, Number 3, July 1999. http://www.acm.org/sigcomm/ccr/archive/ccr-toc/ccr-toc-99.html

[BS00] Balakrishnan, H. and S. Seshan, "The Congestion Manager", RFC 3124, June 2001.

[BV97] S. Biaz and N. Vaidya, "Using End-to-end Statistics to Distinguish Congestion and Corruption Losses: A Negative Result," Texas A&M University, Technical Report 97-009, August 18, 1997.

[BV98] S. Biaz and N. Vaidya, "Sender-Based heuristics for Distinguishing Congestion Losses from Wireless Transmission Losses," Texas A&M University, Technical Report 98-013, June 1998.

[BV98a] S. Biaz and N. Vaidya, "Discriminating Congestion Losses from Wireless Losses using Inter-Arrival Times at the Receiver," Texas A&M University, Technical Report 98-014, June 1998.

[MOGUL] "The Case for Persistent-Connection HTTP", J. C. Mogul, Research Report 95/4, May 1995. Available as http://www.research.digital.com/wrl/techreports/abstracts/95.4.html

[MV97] M. Mehta and N. Vaidya, "Delayed Duplicate-Acknowledgements: A Proposal to Improve Performance of TCP on Wireless Links," Texas A&M University, December 24, 1997. Available at http://www.cs.tamu.edu/faculty/vaidya/mobile.html

[PILC-WEB] http://pilc.grc.nasa.gov/

[PFTK98] Padhye, J., Firoiu, V., Towsley, D. and J.Kurose, "TCP Throughput: A simple model and its empirical validation", SIGCOMM Symposium on Communications Architectures and Protocols, August 1998.

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

[RFC2821] Klensin, J., Editor, "Simple Mail Transfer Protocol", RFC 2821, April 2001.

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

[RFC1191] Mogul J., and S. Deering, "Path MTU Discovery", RFC 1191, November 1990.

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

[RFC2018] Mathis, M., Mahdavi, J., Floyd, S. and A. Romanow "TCP Selective Acknowledgment Options", RFC 2018, October 1996.

[RFC2140] Touch, J., "TCP Control Block Interdependence", RFC 2140, April 1997.

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

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

[RFC2488] Allman, M., Glover, D. and L. Sanchez. "Enhancing TCP Over Satellite Channels using Standard Mechanisms", BCP 28, RFC 2488, January 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.

[RFC2861] Handley, H., Padhye, J. and S., Floyd, "TCP Congestion Window Validation", RFC 2861, June 2000.

[RFC2883] Floyd, S., Mahdavi, M., Mathis, M. and M. Podlosky, "An Extension to the Selective Acknowledgement (SACK) Option for TCP", RFC 2883, August 1999.

[RFC2923] Lahey, K., "TCP Problems with Path MTU Discovery", RFC 2923, September 2000.

[RFC3042] Allman, M., Balakrishnan, H. and S. Floyd, "Enhancing TCP's Loss Recovery Using Limited Transmit", RFC 3042, January, 2001.

[RFC3135] Border, J., Kojo, M., Griner, J., Montenegro, G. and Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations", RFC 3135, June 2001.

[VJ-DCAC] Jacobson, V., "Dynamic Congestion Avoidance / Control" e-mail dated February 11, 1988, available from http://www.kohala.com/~rstevens/vanj.88feb11.txt

[VMPM99] N. Vaidya, M. Mehta, C. Perkins, and G. Montenegro, "Delayed Duplicate Acknowledgements: A TCP-Unaware Approach to Improve Performance of TCP over Wireless," Technical Report 99-003, Computer Science Dept., Texas A&M University, February 1999. Also, to appear in Journal of Wireless Communications and Wireless Computing (Special Issue on Reliable Transport Protocols for Mobile Computing).

## Authors' Addresses

Questions about this document may be directed to:

Spencer Dawkins
Fujitsu Network Communications
2801 Telecom Parkway
Richardson, Texas 75082

Phone: +1-972-479-3782
EMail: spencer.dawkins@fnc.fujitsu.com

Gabriel E. Montenegro
Sun Microsystems
Laboratories, Europe
29, chemin du Vieux Chene
38240 Meylan
FRANCE

Phone: +33 476 18 80 45
EMail: gab@sun.com

Markku Kojo
Department of Computer Science
University of Helsinki
P.O. Box 26 (Teollisuuskatu 23)
FIN-00014 HELSINKI
Finland

Phone: +358-9-1914-4179
EMail: kojo@cs.helsinki.fi

Vincent Magret
Alcatel Internetworking, Inc.
26801 W. Agoura road
Calabasas, CA, 91301

Phone: +1 818 878 4485
EMail: vincent.magret@alcatel.com

Nitin H. Vaidya
458 Coodinated Science Laboratory, MC-228
1308 West Main Street
Urbana, IL 61801

Phone: 217-265-5414
E-mail: nhv@crhc.uiuc.edu

## Full Copyright Statement



## Acknowledgement

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