Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: February 27, 2012

Jim Gettys
Alcatel-Lucent Bell Labs
August 26, 2011

IW10 Considered Harmful
draft-gettys-iw10-considered-harmful-00

Abstract

The proposed change to the initial window to 10 indraft-ietf-tcpm-initcwnd must be considered deeply harmful; not because it is the proposed change is evil taken in isolation, but that other changes in web browsers and web sites that have occurred over the last decade, it makes the problem of transient congestion at a user's broadband connection two and a half times worse. This result has been hidden by the already widespread bufferbloat present in broadband connections. Packet loss in isolation is no longer a useful metric of a path's quality. The very drive to improve latency of web page rendering is already destroying other low latency applications, such as VOIP and gaming, and will prevent reliable rich web real time web applications such as those contemplated by the IETF rtcweb working group.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on February 27, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal

Provisions Relating to IETF Documents
(http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 3
2. Discussion 3
3. Solutions 6
4. IANA Considerations 7
5. Security Considerations 8
6. Informative References 8
Author’s Address 9

1. Introduction

In the second half of the 2000's, competition in web browsers reappeared and changed focus from strictly features, to speed (meaning latency, at least as seen from data-centers, which can be highly misleading), with the discovery (most clearly understood by Google) that web sites are stickier the faster (lower latency) they are. Perhaps Sergey Brin and Larry Page knew Stuart Cheshire at Stanford? [Cheshire].

The problem, in short, is the multiplicative effect of the following:

- Browsers ignoring RFC 2068 [RFC2068] and RFC 2616 [RFC2616] requirement to use no more than two simultaneous TCP connections, with current browsers often using 6 or sometimes many more simultaneous TCP connections

- "Sharded" web sites that sometimes deliberately hide the path to servers actually located in the same data center, to encourage browsers to use even more simultaneous TCP connections

- The proposed change to the TCP congestion window, to allow each fresh TCP connection to send as much as 2.5 times as much data as in the past.

- Current broadband connections having a single queue available to customers, which is usually badly over-buffered, hiding packet loss

- Web pages having large numbers of embedded objects in a web page.

- Web servers having large memory caches and processing power when generating objects on the fly that responses are often/usually transmitted effectively instantaneously at line rate speed

The result can easily be a horrifying large impulse of packets sent effectively simultaneously to the user as a continuous packet train, landing in, and clogging the one queue in their broadband connection and/or home router for extended periods of time. Any chance for your VOIP call to work correctly, or to avoid being fragged in your game, evaporates.

2. Discussion

The original reasons for the 2 TCP connection rule in RFC 2068 [RFC2068] and RFC 2616 [RFC2616] in section 8.1.4 are long gone. In the 1990's era, dial-up modem banks were often badly underbuffered,

and multiple simultaneous connections could easily cause excessive packet loss due to self-congestion either on the dialup port itself, or the dialup bank overall.

Since the 1990’s memory has become very cheap, and we now have the opposite problem: buffering in broad band equipment is much, much too large, larger than any sane amount, as shown by the Netalyzr [Netalyzr] and FCC data [Sundaresan], a phenomena I christened "bufferbloat" as we had lacked a good term for this problem.

What is more, broadband equipment usually provides only a single queue unmanaged, bloated queue to the subscriber; a large impulse of packets for a single user will block other packets from other applications of that user, or to other users who share that connection. This buffering is so large that slow start is badly damaged (TCP will attempt to run many times faster than it should until packet loss finally brings it back under control), and congestion avoidance is no longer stable, as I discovered in 2010 [Gettys].

I had expected and hoped that high performance would be achieved via HTTP Pipelining [HTTPPerf] and that web traffic would have have longer TCP sessions. HTTP pipelining is painful due to HTTP’s lack of any multiplexing layer and the lack of response numbering to allow out of order responses; "poor man’s multiplexing" is possible, but complex. The benefits of pipelining to the length of TCP sessions are somewhat less than one might naively presume, as significantly fewer packets are ultimately necessary. But HTTP pipelining has never seen widespread browser deployment (though is supported by a high fraction of web servers). You will seldom see packet loss by using many TCP connections simultaneously in today’s Internet, as buffers are now so large they can absorb huge transients, sometimes even a megabyte or more.

Web browsers (seeing no packet loss) changed from obeying the RFC2616 requirement of two TCP connections to using 6 or even 15 TCP connections from a browser to a web server. What is more, some web sites, called "sharded web sites," deliberately split themselves across multiple names to trick web browsers into even more profligate uses of TCP connections, and there is no way for a web client to easily determine a web site has been so engineered.

A browser may then render a page with many embedded objects (e.g. images). Current web browsers will therefore simultaneously open or reuse 6 or often many more TCP connections at once, and the initial congestion window of 4 packets may be sent from the same data center simultaneously to a user’s broadband connection. These packets currently enter the single queue in most broadband systems between

the Internet and the home, and with no QOS or other fair queuing present, induce transient latency; your VOIP or gaming packet will be stuck behind this burst of web traffic until the burst drains. Similarly, current home routers often lack any QOS or sophisticated queuing to ensure fairness between different users. The proposal by Chu, et. al. in to raise the initial congestion window from four to 10, making the blocking and resulting latency and jitter problem up to 2.5 times worse.

Note that broadband equipment is not the only overbuffered equipment in most user's paths. Home routers, 3g wireless and the user's operating systems are usually even worse than broadband equipment. In a user's home, whenever the wireless bandwidth happens to be below that of the broadband connection, the bottleneck link is in the wireless hop, and so the problem may occur there rather than in the broadband connection. It is the bottleneck transfer that matters; not the theoretical bandwidth of the links. 802.11g is at best 20Mbps, but often much worse. Other bottleneck points in the user's paths may also be lacking AQM.

I believe the performance analysis in the draft-ietf-tcpm-initcwnd is flawed not by being incorrect in what it presents, but by overlooking the latency and jitter inflicted on other traffic that is sharing the broadband link, due to the large buffering in these links and typically single queue. It is the damage the IW change would make to other real time applications sharing that link (including rtcweb applications), or what those sharing that link do to you that is the issue.

Simple arithmetic to compute induced transient latency, even ignoring all overhead, comes up with scary results:

Latency table

# conn	ICW=4 (bytes)	Time @1Mbps	Time @50Mbps	ICW=10	Time @1Mbps	Time @50Mbps
2	12000	96ms	1.92ms	30000	240ms	4.8ms
6	36000	288ms	5.76ms	90000	720ms	14ms
15	90000	720ms	14.4ms	225000	1800ms	36ms
30	180000	1440ms	28.8ms	450000	3600ms	72ms

Table 1: Unloaded Latency

1 Mbps may be your fair share of a loaded 802.11 link. 50Mbps is near the top end of today's broadband. Available bandwidths in other

parts of the world are often much, much lower than in parts of the world where broadband has deployed.

Simple experiments over 50Mbps home cable service against Google images confirm latencies that reach or sometimes double those in the table. Steady-state competing TCP traffic will multiply these times correspondingly; even at 50Mbps, reliable, low latency VOIP can therefore be problematic. From this table, it becomes obvious that QOS in shared wireless networks has become essential, if only because of this change in web browser behavior. Note that the 2 connection rule still results in 100ms latencies on 1Mbps connections, which is already very problematic for VOIP by induction of jitter. Two TCP connections are capable of driving a megabit link at saturation over most paths today even from a cold start with ICW=4.

In the effort to maximise speed (as seen by a data center) web browsers/servers have turned web traffic into an delta function congesting the user’s queue for extended periods. Since the broadband edge is badly over-buffered as shown first by Netalyzr, packets are usually not lost, but instead, fill the one queue separating people from the rest of the Internet until they drain.

Many carrier’s telephony services are not blocked by this web traffic since the carriers have generally provisioned voice channels independently of data service; but most competing services such as Vonage or Skype will be blocked, as they must use the single, oversized queue. While I do not believe this advantage was by design, it is an effect of bufferbloat and current broadband supporting only a single queue, at most accelerating acks ahead of other bulk data packets. In the presently deployed broadband infrastructure, these other queues are usually unavailable for use by time sensitive traffic, and DiffServ [RFC3260] is not implemented in broadband head end equipment. Therefore time sensitive packets share the same queue of non-time sensitive bulk data (HTTP) traffic.

3. Solutions

If HTTP pipelining were deployed, it would result in lower actual times to most users; fewer bytes are needed due to sharing packets among objects and requests, and much lower packet overhead and lower ack traffic and significantly better TCP congestion behavior. While increasing the initial window someday may indeed make sense, it is truly a frightening to us to raise the ICW during this arms race given already deployed HTTP/1.1 implementations. SPDY [SPDY] should have similar (or better) results, but requires server side support that will take time to develop and deploy, whereas most deployed web servers have supported pipelining for over a decade (sometimes with

bugs, which is part of why it is painful to deploy web client HTTP pipelining).

A full discussion of solutions that would improve latency for general web browsing without destroying realtime applications is beyond the scope of this ID. I note a few quickly (which are not mutually exclusive) that can and should be persued. They all have differing time scales and costs; all are desirable in my view, but the discussion would be much more than I can cover here.

- o Deployment of HTTP/1.1 pipelining (with reduction of # of simultaneous connections back to RFC 2616 levels

- o Deployment of SPDY

- o DiffServ deployment in the broadband edge and its use by applications

- o DiffServ deployment in home routers (which often unbeknownst to those not in the gaming industry, has already partially occurred due to its inclusion in the default Linux PFIFO_FAST line discipline).

- o Some sort of "per user" or "per machine" queuing mechanism on broadband connections such that complete starvation of service for extended periods can be avoided.

At a deeper and more fundamental level, individual applications (such as web browsers) may game the network with concurrent bad results to the user (or other users sharing that edge connection), and with the advent of Web sockets, even individual web applications may similarly game the network’s behavior. With the huge dynamic range of today’s edge environments, we have no good way to know what a "safe" initial impulse of packets a server may send into the network in what situation. Today there is no disincentive to applications abusing the network. Congestion exposure mechanisms such as Congestion Exposure [ConEx] are badly needed, and some way to enable users (and their applications, on their behalf) to be aware of and react to badly behaved applications.

4. IANA Considerations

This memo includes no request to IANA.

5. Security Considerations

Current practice of web browsers, in concert with "sharded" web sites and changes to the initial congestion window, and the currently deployed broadband infrastructure can be considered a denial of (low latency) service attack on consumer's broadband service.

6. Informative References

[Cheshire] Cheshire, "It's the Latency, Stupid", 1996, <http://rescomp.stanford.edu/~cheshire/rants/Latency.html>.

[Chu] Chu, Dukkipati, Cheng, and Mathis, "Increasing TCP's Initial Window", 2011, <http://datatracker.ietf.org/doc/draft-ietf-tcpm-initcwnd/>.

[ConEx] Briscoe, "congestion exposure (ConEx), re-feedback and re-ECN", 2005, <http://www.bobbriscoe.net/projects/refb/>.

[Gettys] Gettys, "Whose house is of glasse, must not throw stones at another", January 2011, <http://gettys.wordpress.com/2010/12/06/whose-house-is-of-glasse-must-not-throw-stones-at-another/>.

[HTTPPerf] Nielsen, Gettys, Baird-Smith, Prud'hommeaux, Lie, and Lilley, "Network Performance Effects of HTTP/1.1, CSS1, and PNG", June 1997, <http://www.w3.org/Protocols/HTTP/Performance/Pipeline.html>.

[Netalyzr] Kreibich, Weaver, Nechaev, and Paxson, "Netalyzr: illuminating the edge network.", November 2010, <http://www.icir.org/christian/publications/2010-imc-netalyzr.pdf>.

[RFC2068] Fielding, R., Gettys, J., Mogul, J., Nielsen, H., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2068, January 1997.

[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.

[RFC3260] Grossman, D., "New Terminology and Clarifications for Diffserv", RFC 3260, April 2002.

[SPDY] Belshe, "SPDY: An experimental protocol for a faster web", 2011, <http://dev.chromium.org/spdy>.

[Sundaresan] Sundaresan, de Donato, Feamster, Teixeira, Crawford, and Pescape, "Broadband Internet Performance: A View From the Gateway, Proceedings of SIGCOMM 2011", August 2011, <http://gtnoise.net/papers/2011/sundaresan:sigcomm2011.pdf>.

Author's Address

Jim Gettys
Alcatel-Lucent Bell Labs
21 Oak Knoll Road
Carlisle, Massachusetts 01741
USA

Phone: +1 978 254-7060
Email: jg@freedesktop.org