Responsiveness under Working Conditions
draft-cpaasch-ippm-responsiveness-00

Abstract

   For many years, a lack of responsiveness, variously called _lag_,
   latency, or _bufferbloat,_ has been recognized as an unfortunate,
   but common symptom in today's networks. Even after a decade of work
   on standardizing technical solutions, it remains a common problem for
   the end users.

   Everyone "knows" that it is "normal" for a video conference to have
   problems when somebody else at home is watching a 4K movie or
   uploading photos from their phone. However, there is no technical
   reason for this to be the case. In fact, various queue management
   solutions (fq_codel, cake, PIE) have solved the problem for tens of
   thousands of people.

   Our networks remain unresponsive, not from a lack of technical
   solutions, but rather a lack of awareness of the problem. We believe
   that creating a tool whose measurement matches people's every day
   experience will create the necessary awareness, and result in a
   demand for products that solve the problem.

   This document specifies the "RPM Test" for measuring responsiveness.
   It uses common protocols and mechanisms to measure user experience
   especially when the network is fully loaded ("responsiveness under
   working conditions".) The measurement is expressed as "Round-trips
   Per Minute" (RPM) and should be included with throughput (up and
   down) and idle latency as critical indicators of network quality.

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 https://datatracker.ietf.org/drafts/current/.

Table of Contents

1. Introduction

For many years, a lack of responsiveness, variously called _lag_,
latency, or _bufferbloat,_ has been recognized as an unfortunate,
but common symptom in today's networks [Bufferbloat]. Solutions like
fq_codel [RFC8289] or PIE [RFC8033] have been standardized and are to
some extent widely implemented. Nevertheless, people still suffer
from bufferbloat.

Although significant, the impact on user experience can be transitory
- that is, its effect is not always present. Whenever a network is
actively being used at its full capacity, buffers can fill up and
create latency for traffic. The duration of those full buffers may
be brief: a medium-sized file transfer, like an email attachment or
uploading photos, can create bursts of latency spikes. An example of
this is lag occurring during a videoconference, where a connection is
briefly shown as unstable.

These short-lived disruptions make it hard to narrow down the cause.
We believe that it is necessary to create a standardized way to
measure and express responsiveness.

Existing network measurement tools could incorporate a responsiveness
measurement into their set of metrics. Doing so would also raise the
awareness of the problem and make the standard "network quality
measures" of throughput, idle latency, and responsiveness.

1.1. Terminology

A word about the term "bufferbloat" - the undesirable latency that
comes from a router or other network equipment buffering too much
data. (1) This document uses the term as a general description of bad
latency, using more precise wording where warranted.

"Latency" is a poor measure of responsiveness, since it can be hard
for the general public to understand. The units are unfamiliar
("what _is_ a millisecond?") and counterintuitive ("100 msec - that
sounds good - it's only a tenth of a second!").

Instead, we create the term "Responsiveness under working conditions"
to make it clear that we are measuring all, not just idle,
conditions, and use "round-trips per minute" as the metric. The
values range from 50 (poor) to 3,000 (excellent), with the added
advantage that "bigger is better." Finally, we abbreviate the
measurement to "RPM", a wink to the "revolutions per minute" that we
use for cars.

This document defines an algorithm for the "RPM Test" that explicitly measures responsiveness under working conditions.

2. Design Constraints

There are many challenges around measurements on the Internet. They include the dynamic nature of the Internet, the diverse nature of the traffic, the large number of devices that affect traffic, and the difficulty of attaining appropriate measurement conditions.

Internet paths are changing all the time. Daily fluctuations in the demand make the bottlenecks ebb and flow. To minimize the variability of routing changes, it's best to keep the test duration relatively short.

TCP and UDP traffic, or traffic on ports 80 and 443, may take significantly different paths on the Internet and be subject to entirely different Quality of Service (QoS) treatment. A good test will use standard transport layer traffic – typical for people's use of the network – that is subject to the transport's congestion control that might reduce the traffic's rate and thus its buffering in the network.

Traditionally, one thinks of bufferbloat happening on the routers and switches of the Internet. However, the networking stacks of the clients and servers can have huge buffers. Data sitting in TCP sockets or waiting for the application to send or read causes artificial latency, and affects user experience the same way as "traditional" bufferbloat.

Finally, it is important to note that queueing only happens behind a slow "bottleneck" link in the network, and only occurs when sufficient traffic is present. The RPM Test must ensure that buffers are actually full for a sustained period, and only then make repeated latency measurements in this particular state.

3. Goals

The algorithm described here defines an RPM Test that serves as a good proxy for user experience. This means:

1. Today's Internet traffic primarily uses HTTP/2 over TLS. Thus, the algorithm should use that protocol.

    As a side note: other types of traffic are gaining in popularity (HTTP/3) and/or are already being used widely (RTP). Traffic prioritization and QoS rules on the Internet may subject traffic to completely different paths:

these could also be measured separately.

2. The Internet is marked by the deployment of countless middleboxes
    like transparent TCP proxies or traffic prioritization for
    certain types of traffic. The RPM Test must take into account
    their effect on DNS-request, TCP-handshake, TLS-handshake, and
    request/response.

3. The test result should be expressed in an intuitive, nontechnical
    form. People commonly look for a single "score" of their
    performance.

4. Finally, to be useful to a wide audience, the measurement should
    finish within a short time frame. Our target is 20 seconds.

4. Measuring Responsiveness Under Working Conditions

   To make an accurate measurement, the algorithm must reliably put the
   network in a state that represents those "working conditions". Once
   the network has reached that state, the algorithm can measure its
   responsiveness. The following explains how the former and the latter
   are achieved.

4.1. Working Conditions

   For the purpose of this methodology, typical "working conditions"
   represent a state of the network in which the bottleneck node is
   experiencing ingress and egress flows similar to those created by
   humans in the typical day-to-day pattern.

   While a single HTTP transaction might briefly put a network into
   working conditions, making reliable measurements requires maintaining
   the state over sufficient time.

   The algorithm must also detect when the network is in a persistent
   working condition, also called "saturation".

   Desired properties of "working condition":

   * Should not waste traffic, since the person may be paying for it

   * Should finish within a short time to avoid impacting other people
      on the same network, to avoid varying network conditions, and not
      try the person's patience.

4.1.1. From single-flow to multi-flow

   A single TCP connection may not be sufficient to saturate a path.
   For example, the 4MB constraints on TCP window size constraints may
   not fill the pipe. Additionally, traditional loss-based TCP
   congestion control algorithms react aggressively to packet loss by
   reducing the congestion window. This reaction (intended by the
   protocol design) decreases the queueing within the network, making it
   hard to reach saturation.

   The goal of the RPM Test is to keep the network as busy as possible
   in a sustained and persistent way. It uses multiple TCP connections
   and gradually adds more TCP flows until saturation is reached.

4.1.2. Parallel vs Sequential Uplink and Downlink

   Poor responsiveness can be caused by queues in either (or both) the
   upstream and the downstream direction. Furthermore, both paths may
   differ significantly due to access link conditions (e.g., 5G
   downstream and LTE upstream) or the routing changes within the ISPs.
   To measure responsiveness under working conditions, the algorithm
   must saturate both directions.

   Measuring in parallel achieves more data samples for a given
   duration. Given the desired test duration of 20 seconds, sequential
   uplink and downlink tests would only yield half the data. The RPM
   Test specifies parallel, concurrent measurements.

   However, a number caveats come with measuring in parallel:

   * Half-duplex links may not permit simultaneous uplink and downlink
      traffic. This means the test might not saturate both directions
      at once.

   * Debuggability of the results becomes harder: During parallel
      measurement it is impossible to differentiate whether the observed
      latency happens in the uplink or the downlink direction.

   * Consequently, the test should have an option for sequential
      testing.

4.1.3. Reaching saturation

   The RPM Test gradually increases the number of TCP connections and
   measures "goodput" - the sum of actual data transferred across all
   connections in a unit of time. When the goodput stops increasing, it
   means that saturation has been reached.

Saturation has two criteria: a) the load bearing connections are utilizing all the capacity of the bottleneck, b) the buffers in the bottleneck are completely filled.

The algorithm notes that throughput gradually increases until TCP connections complete their TCP slow-start phase. At that point, throughput eventually stalls usually due to receive window limitations. The only means to further increase throughput is by adding more TCP connections to the pool of load bearing connections. If new connections leave the throughput the same, saturation has been reached and - more importantly - the working condition is stable.

Filling buffers at the bottleneck depends on the congestion control deployed on the sender side. Congestion control algorithms like BBR may reach high throughput without causing queueing because the bandwidth detection portion of BBR effectively seeks the bottleneck capacity.

RPM Test clients and servers should use loss-based congestion controls like Cubic to fill queues reliably.

The RPM Test detects saturation when the observed goodput is not increasing even as connections are being added, or it detects packet loss or ECN marks signaling congestion or a full buffer of the bottleneck link.

4.1.4. Final "Working Conditions" Algorithm

The following algorithm reaches working conditions (saturation) of a network by using HTTP/2 upload (POST) or download (GET) requests of infinitely large files. The algorithm is the same for upload and download and uses the same term "load bearing connection" for each.

The steps of the algorithm are:

*   Create 4 load bearing connections

*   At each 1 second interval:

    -   Compute "instantaneous aggregate" goodput which is the number of bytes transferred within the last second.

    -   Compute a moving average of the last 4 "instantaneous aggregate goodput" measurements

    -   If moving average > "previous" moving average + 5%:

   o Network did not yet reach saturation. If no flows added
    within the last 4 seconds, add 4 more flows

  - Else, network reached saturation for the current flow count.

   o If new flows added and for 4 seconds the moving average
    throughput did not change: network reached stable saturation

   o Else, add four more flows

Note: It is tempting to envision an initial base RTT measurement and
adjust the intervals as a function of that RTT. However, experiments
have shown that this makes the saturation detection extremely
unstable in low RTT environments. In the situation where the
"unloaded" RTT is in the single-digit millisecond range, yet the
network's RTT increases under load to more than a hundred
milliseconds, the intervals become much too low to accurately drive
the algorithm.

(I'm not sure what this last caveat re: low RTT means in practice)

4.2. Measuring Responsiveness

Once the network is in a consistent working conditions, the RPM Test
must "probe" the network multiple times to measure its
responsiveness.

Each RPM Test probe measures:

1. The responsiveness of the different steps to create a new
    connection, all during working conditions.

    To do this, the test measures the time needed to make a DNS
    request, establish a TCP connection on port 443, establish a TLS
    context using TLS1.3, and send and receive a one-byte object with
    a HTTP2 GET request. It repeats these steps multiple times for
    accuracy.

2. The responsiveness of the network and the client/server
    networking stacks for the load bearing connections themselves.

    To do this, the load bearing connections multiplex an HTTP/2 GET
    request for a one-byte object to get the end-to-end latency on
    the connections that are using the network at full speed.

4.2.1. Aggregating the Measurements

   The algorithm produces sets of 5 times for each probe, namely: DNS
   handshake, TCP handshake, TLS handshake, HTTP/2 request/response on
   separate (idle) connections, HTTP/2 request/response on load bearing
   connections. This fine-grained data is useful, but not necessary for
   creating a useful metric.

   To create a single "Responsiveness" (e.g., RPM) number, this first
   iteration of the algorithm gives an equal weight to each of these
   values. That is, it sums the five time values for each probe, and
   divides by the total number of probes to compute an average probe
   duration. The reciprocal of this, normalized to 60 seconds, gives
   the Round-trips Per Minute (RPM).

4.2.2. Statistical Confidence

   The number of probes necessary for statistical confidence is an open
   question. One could imagine a computation of the variance and
   confidence interval that would drive the number of measurements and
   balance the accuracy with the speed of the measurement itself.

   It is also an open topic to compare the RPM Test to other standard
   measurement tools such as DSLReports Speed Test (2) the Waveform
   Bufferbloat test (3); the command-line betterspeedtest.sh (4)

5. RPM Test Server API

   The RPM measurement uses standard protocols: no new protocol is
   defined.

   Both the client and the server MUST support HTTP/2 over TLS 1.3. The
   client MUST be able to send a GET request and a POST. The server
   MUST be able to respond to both of these HTTP commands. Further, the
   server endpoint MUST be accessible through a hostname that can be
   resolved through DNS. The server MUST have the ability to provide
   content upon a GET request. Both client and server SHOULD use loss-
   based congestion controls like Cubic. The server MUST use a packet
   scheduling algorithm that minimizes internal queueing to avoid
   affecting the client's measurement.

   The server MUST respond to 4 URLs:

1. A "small" URL/response: The server must respond with a status
    code of 200 and 1 byte in the body. The actual body content is
    irrelevant.

2. A "large" URL/response: The server must respond with a status
    code of 200 and a body size of at least 8GB. The body can be
    bigger, and may need to grow as network speeds increases over
    time. The actual body content is irrelevant. The client will
    probably never completely download the object, but will instead
    close the connection after reaching working condition and making
    its measurements.

3. An "upload" URL/response: The server must handle a POST request
    with an arbitrary body size. The server should discard the
    payload.

4. A configuration URL that returns a JSON object with the
    information the client uses to run the test (sample below). All
    the fields are required except "test_endpoint". Sample JSON:

```
{
  "version": 1,
  "urls": {
    "small_https_download_url": "https://example.apple.com/api/v1/small",
    "large_https_download_url": "https://example.apple.com/api/v1/large",
    "https_upload_url": "https://example.apple.com/api/v1/upload"
  },
  "test_endpoint": "hostname123.cdnprovider.com"
}
```

    If the "test_endpoint" field is present, it is an indication that the
    Service provider/content distribution network (CDN) is able to "pin"
    all of the requests for a particular test run to a specific server.
    The client should look up the test_endpoint name and use the
    resulting address as the host for all the other URLs. A CDN should
    supply a test_endpoint so that measurements use the same server/
    follow the same paths to avoid switching servers during a test run.

    The client begins the responsiveness measurement by querying for the
    JSON configuration. This supplies the URLs for creating the load
    bearing connections in the upstream and downstream direction as well
    as the small object for the latency measurements.

6. Reference Implementations

    The RPM Test has been implemented in the following settings:

    * macOS 15

    * iOS

    * Python?

   * Javascript?

7. Security Considerations

   TBD

8. IANA Considerations

   TBD

9. Acknowledgments

   * Dave Taht - Godfather of Bufferbloat research, Creator of CeroWrt,
      the testbed for testing fq_codel and solving Bufferbloat

   * Rich Brown - Editorial pass over this I-D

10. References

   * (1) Bufferbloat definition: (https://www.bufferbloat.net/
      projects/)

   * (2) DSLReports Speed Test: (http://DSLReports.com/speedtest)

   * (3) Waveform Bufferbloat Test; (https://www.waveform.com/tools/
      bufferbloat)

   * (4) betterspeedtest.sh (https://github.com/richb-
      hanover/OpenWrtScripts/blob/master/betterspeedtest.sh)

11. Informative References

   [Bufferbloat]
              Gettys, J. and K. Nichols, "Bufferbloat: Dark Buffers in
              the Internet", Communications of the ACM, Volume 55,
              Number 1 (2012) , n.d..

   [RFC8033] Pan, R., Natarajan, P., Baker, F., and G. White,
              "Proportional Integral Controller Enhanced (PIE): A
              Lightweight Control Scheme to Address the Bufferbloat
              Problem", RFC 8033, DOI 10.17487/RFC8033, February 2017,
              <https://www.rfc-editor.org/info/rfc8033>.

   [RFC8289] Nichols, K., Jacobson, V., McGregor, A., Ed., and J.
              Iyengar, Ed., "Controlled Delay Active Queue Management",
              RFC 8289, DOI 10.17487/RFC8289, January 2018,
              <https://www.rfc-editor.org/info/rfc8289>.

Authors' Addresses

    Christoph Paasch
    Apple Inc.
    One Apple Park Way
    Cupertino, California 95014,
    United States of America

    Email: cpaasch@apple.com

    Randall Meyer
    Apple Inc.
    One Apple Park Way
    Cupertino, California 95014,
    United States of America

    Email: rrm@apple.com

    Stuart Cheshire
    Apple Inc.
    One Apple Park Way
    Cupertino, California 95014,
    United States of America

    Email: cheshire@apple.com

    Omer Shapira
    Apple Inc.
    One Apple Park Way
    Cupertino, California 95014,
    United States of America

    Email: oesh@apple.com