Preliminary Longitudinal Study of Internet Responsiveness

Matt Mathis, Aug 2021

This is a very preliminary longitudinal study of Internet responsiveness based on data collected by Measurement Lab (MLab). It is quite limited in a number of ways, none-the-less it shows improvement of Internet Responsiveness over the several years. However, the primary contributions of this note are insights into some of the open questions and work that needs to be completed in order to develop a well specified Internet responsiveness metric.

This study is based on a speculative approximation of a future formal "Responsiveness Under Working Conditions" metric, patterned after a developer measurement tool that Apple is already shipping with iPhone and iOS devices.[Apple] Although there is a partial draft specification [Draft] both implementations long precede any formal specification and validation.

Responsiveness is tentatively defined as the rate for round trips in the network, typically as Rounds per Minute (RPM). Since queues in the network are known to be a major contributor to poor responsiveness [BufferBloat], the metric of interest is actually "Responsiveness Under Working Conditions" which requires an authentic load on the measurement stream, and potentially additional network cross traffic to expose the behavior of the queue management at potential bottlenecks.

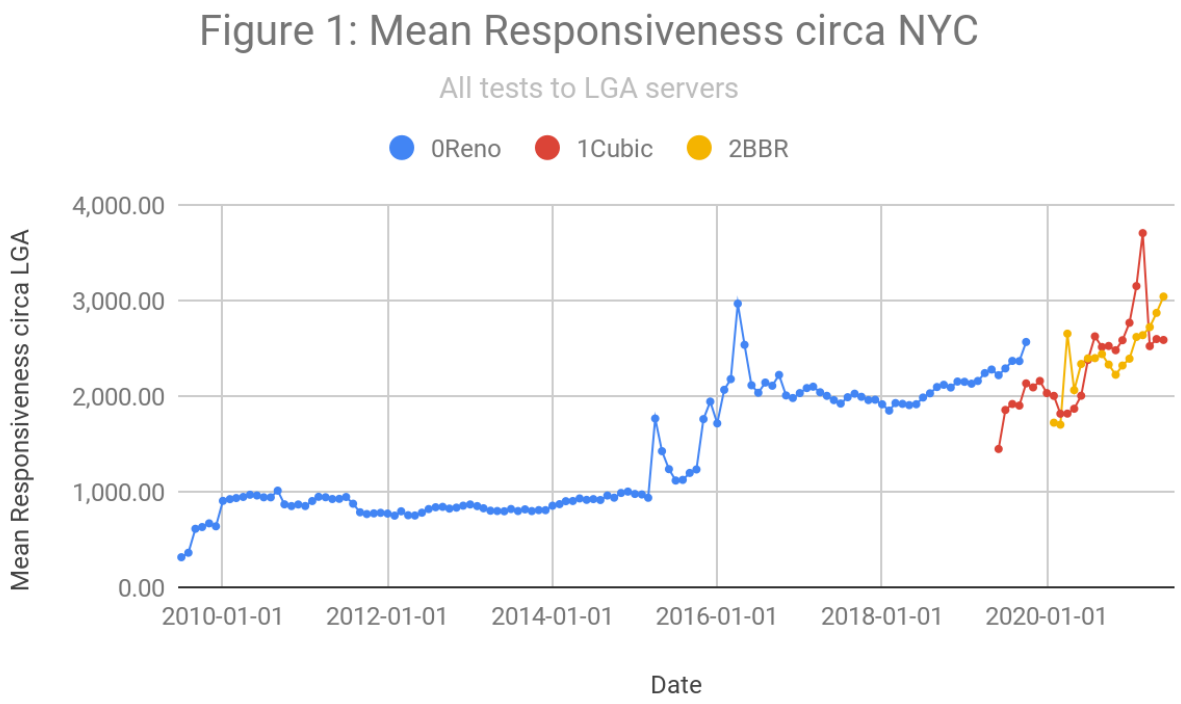
It is expected that we will ultimately define parallel responsiveness metrics at different levels in the stack. For this paper we estimate counting transport layer rounds: data segments, trigger ACKs which in turn trigger additional segments. The Apple tool measures application rounds, by using application layer ping messages to measure the Round Trip Time of application to application messages. In addition to network delays, application messages must traverse all of the queues and buffering in the operating system as well as the application itself.

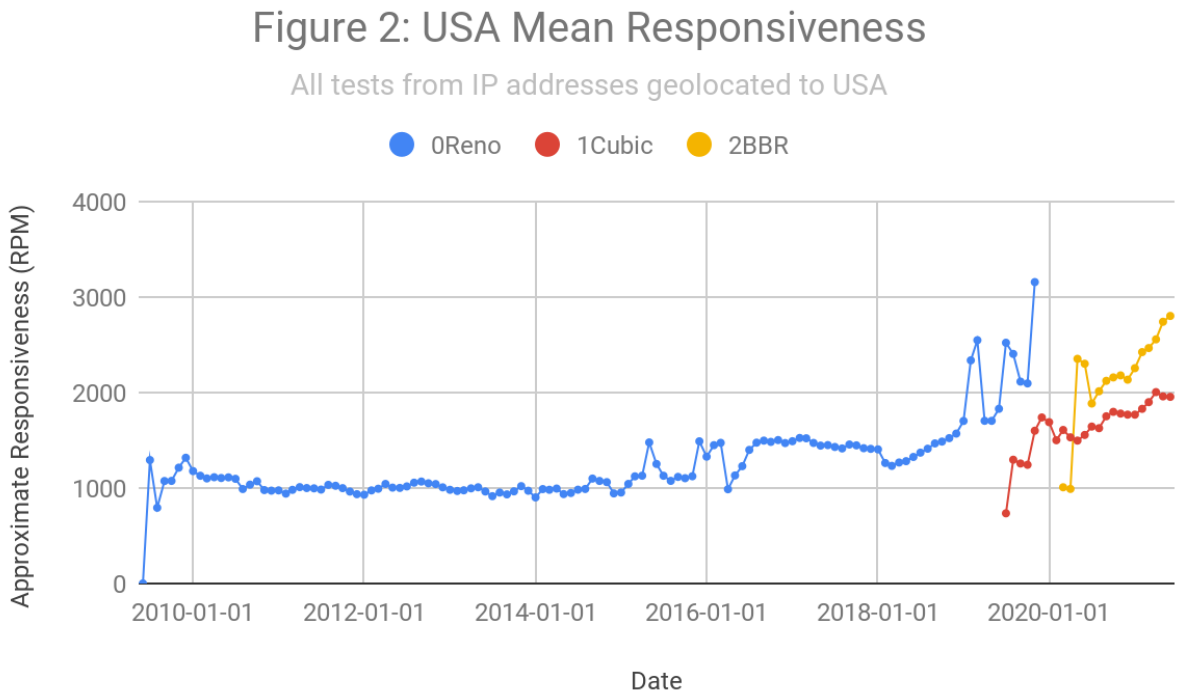
The metric presented in this note is computed from the final value of the Smoothed RTT estimator maintained by TCP. The final value is not an ideal method for estimating responsiveness over the lifetime of the connection, but it is good enough to get some insight into both the evolution of the Internet and considerations for specifying future formal responsiveness metrics.

The MLab archive includes measurements from more than 4 billion download tests. Every one of them includes periodic TCP state snapshots via Web100 or TCPinfo. All of the snapshots

are preserved in the raw data in Google Cloud Storage (generally at 10 mS intervals). However this data was downsampled as it was parsed into BigQuery. TCPinfo data from the new platform was thinned to 100 mS samples, but the Web100 data only preserved the final snapshot.

The data presented in Figures 1 and 2 is very preliminary, without any filtering or grooming. Every NDT connection that reached the TCP established state is included in the plot, even if it is otherwise not a meaningful measurement. Appendix A includes the SQL and instructions for reproducing these plots.





Observations and future work

There are discontinuities in the data that correspond to MLab measurement platform upgrades. We partitioned the data by TCP Congestion Control and implicitly platform version, because we expect CC to have some effect on network queue occupancy and thus responsiveness. The transition from Reno to Cubic reflects a complete measurement platform upgrade that was rolled out over approximately half a year. Since CUBIC maximizes throughput by maintaining larger network queues, it causes substantially lower responsiveness.

The transition from CUBIC to BBR reflects a much more gradual transition to a complete reimplementation on NDT, that uses the BBR congestion control. BBR is intended to more accurately model the network capacity. In the NYC data they don't appear to be materially different, however for the entire US they clearly are - BBR maintains smaller queues and better responsiveness.

Note that tool discontinuities is an intrinsic problem for any long duration longitudinal data set. An ideal, state-of-the-art measurement tool in 2009 is unlikely to be good enough (and certainly not state-of-the-art) by 2021.

Since this data has not been filtered or groomed in any way it contains measurements that should properly be discarded: connections that did not send significant data (not "working conditions") and even MLab's own OAM traffic. A future version of this note will be based on properly groomed data.

Even without grooming, adjacent months typically exhibit very similar responsiveness, suggesting that the measurements are quite stable and do not have much noise. Nearly all outliers involve more than one month, suggesting that they are caused by real signals in the data. Note that almost all outliers are parallel between the data sets, except circa April 2016. Why? Once we have improved the data grooming, we will look harder at the outliers.

We selected NYC because there is a huge population in a relatively small geographical radius. Most MLab test traffic goes to the closest server. The vernoi map for NYC is bound by Washington DC, Tronto and Montreal, which makes it one of the geographically smallest regions in the entire fleet. This was intended to minimize the extent to which the client geographical distribution blurs other effects such as changes to queue management.

It was very unexpected that the NYC and all US data is so similar. This is probably an indication that the vast majority of tests are from metropolitan areas, each of which has their own servers, and people in the huge geographical expenses of the American west don't run many tests.

We have not looked at the European or Southern Hemisphere data at all, because MLab's footprint changed too much to easily make meaningful comparisons. For most of the US the locations of the "closest servers" have not changed since 2010. This is not true for Europe - to make meaningful comparisons in Europe it will be necessary to place restrictions on both the server and client locations. For example, to exclude old tests which went to different servers than the closest available servers today. Otherwise the old tests will include excess delay due to longer path lengths. Note that a future formal responsiveness metric will need to directly address issues of CDN locations, peering and topology, especially since some of the stated queueing targets are below the speed of light delays for many paths.

As mentioned earlier, the raw MLab data archived in GCS includes periodic TCP state snapshots for every NDT test. We have already initiated the process to reparse all of the raw Web100 data to import periodic snapshots into BigQuery. Once the raw data has been reparsed, we will look at better algorithms for computing responsiveness from the entire SRTT time series for each NDT test.

The smoothed RTT estimator was introduced by Van Jacobson in 1988, and has since been standardized[SRTT]. One of the strengths of using a metric based on SRTT is that it has

changed relatively little over the 30+ years since it was first developed, making it more likely that metrics derived from SRTT values archived on different vintage systems can be compared directly.

MLab also has full TCP header captures from the majority of NDT tests. We hope to calibrate responsiveness metrics by reconstructing protocol timing diagrams and counting round trips directly for some small number of tests. These timing diagrams will be close proxies for the actual ground truth of old measurements.

There is an important corner case that we need to consider: We are aware that there are a significant number of clients that run loss-less because they have oversized buffers at the bottleneck [Bufferbloat]. These clients typically exhibit a large bottleneck queue that is still growing at the end of the test. (e.g. a 5 second queue for a 10 second test). For these clients, the final SRTT (and for that matter any summary of the SRTT timeseries) is likely to depend on the test duration (or possibly the maximum value of the receiver window). How do we properly include such paths in any aggregate responsiveness metric?

Conclusion

We have a lot of work to do. As intended, this study asks far more new questions than it answers.

The good news is that in at least some parts of the internet appear to be more responsive now, than they were a few years ago.

References

[Apple] WWDC21 video "Reduce network delays for your app"
[Draft] Responsiveness under working conditions
[SRTT] RFC 6298, V. Paxson, "Computing TCP's Retransmission Timer"

Appendix A

To reconstruct the graphs follow the procedure on the BigQuery QuickStart page https://www.measurementlab.net/data/docs/bq/quickstart/ .

Then paste the following into the query editor (you may need to convert UTF back into ASCII):

```
# Preliminary Longitudinal Study of Internet Responsiveness
#
# See https://www.measurementlab.net/tests/ndt/ and
# https://www.measurementlab.net/tests/ndt/views/custom/
# for documentation on these tables.

WITH

# Separate Responsiveness metrics for each tool
Web100Responsiveness AS (
   SELECT id, date, a, filter, node, client, server,
   # NB: web100_log_entry.snap is really just the final snap
   _internal202010.web100_log_entry.snap.SmoothedRTT AS FinalSRTT,
   SAFE_DIVIDE(60000.0, _internal202010.web100_log_entry.snap.SmoothedRTT) AS FinalRPM,
   FROM `measurement-lab.ndt_intermediate.extended_web100_downloads`
),
NDT5Responsiveness AS (
SELECT id, date, a, filter, node, client, server,
   _internal202010.FinalSnapshot.TCPInfo.RTT AS FinalSRTT,
   SAFE_DIVIDE(60000000.0,_internal202010.FinalSnapshot.TCPInfo.RTT) AS FinalRPM,
   FROM `measurement-lab.ndt_intermediate.extended_ndt5_downloads`
),
NDT7Responsiveness AS (
SELECT id, date, a, filter, node, client, server,
   _internal202010.lastsample.TCPInfo.RTT AS FinalSRTT,
   SAFE_DIVIDE(60000000.0,_internal202010.lastsample.TCPInfo.RTT) AS FinalRPM,
   FROM `measurement-lab.ndt_intermediate.extended_ndt7_downloads`
),
UnifiedResponsiveness AS (
   SELECT * FROM NDT7Responsiveness
   UNION ALL
   SELECT * FROM NDT5Responsiveness
   UNION ALL
   SELECT * FROM Web100Responsiveness
),

# Example queries of UnifiedResponsiveness
# Edit to make it your own
example AS (
   SELECT
      DATE_TRUNC(date, month) as month,
```

```
      a.CongestionControl,
      AVG(FinalRPM) MeanFinalRPM,
      APPROX_QUANTILES(FinalRPM, 100)[offset(50)] AS MedianFinalRPM,
    FROM UnifiedResponsiveness
    WHERE
      date between '2009-01-01' AND '2021-08-01'
  #     AND site LIKE "lga%"
  #     AND client.Network.ASnumber in ( 7922 )
      AND Client.Geo.CountryCode = "US"
    GROUP BY month, a.CongestionControl
    ORDER BY month, a.CongestionControl
)

SELECT * FROM example
```