NOMA Data Measurements Project Template – *DRAFT*

December 8, 2016.

This document outlines the template for network operators' self-instrumentation and contributions to the NOMA project.

The value of network instrumentation to operators is the creation of a coherent and detailed picture of customers' network experience, facilitating the identification of issues and providing information to pinpoint problem sources for debugging (e.g., internal to their network or not).

The purpose of sharing (sanitized) data is to provide an amalgamated reference view of network experience, within and across geographies.

This is currently very much in draft form as a discussion document. Key points for discussion are about coordination for coverage (website lists, frequency of measurements, percentage of network users reflected).

Measurements framework

Network coverage

Ideally, a network's access endpoints should be completely covered by this measurements framework.

Practically, it can be deployed in stages.

Data collected

Using "`libcurl`" and `http` as the closest approximation of end users' experience, each collection point (as close as possible to the customer)[1]:

[1] Ideal: closest layer 3 device to the customer site; perhaps DSLAM or other aggregator

- IPv6 DNS lookup to each target website
- Time to connect to the target website over IPv6
- Total time for each target website over IPv6
- IPv4 DNS lookup to each target website
- Time to connect to the target website over IPv4
- Total time for each target website over IPv4
- Traceroute to IP address of each target website

Frequency

Each collection point will collect data from websites ?? times per day.

Test range

Each collection point should be configured to test against ?? websites.

On coordination

The sample code in the appendix runs every 60 seconds. If every operator runs this (type of) test every 60 seconds, against the Alexa top 1,000, there might be a noticeable impact.

Network internal use of the data

Internal uses for the collected data will include detecting and resolving internal network issues, as well as connectivity issues to tested websites.

Once the high total time cases have been identified, by using the WLA system, Comcast uses its Traceroute Analysis system to diagnose the issue and identify the root cause of the problem. For example, from Comcast's experience:

> The Traceroute Analysis system database stores details of each probe – website traceroute (including all the traceroute hops), and the Traceroute Analysis system is able to review the past data and identify whether the increased RTT is within the Comcast network or outside, exactly at what hop in the traceroute does the RTT go up, the different servers which served the website and whether high RTT was contributed by only one of those servers.[2]

Shared data for the NOMA project

The NOMA project will publish aggregated data on a geographic regional basis.

TBD: aggregation by collapsing multiple providers' data in one region, averaging across measurement times?

---

[2] http://www.internetsociety.org/sites/default/files/pdf/accepted/9_isoc.comcast_latency_workshop_proposal_final.pdf

Appendix – sample collection code

The following is a simple shell script for collecting (every minute) the data above for a list of fully qualified domain names contained in a file named "`url.list`".

```
#!/bin/sh

echo "# FQDN,IPv6_NL,IPv6_TC,IPv6_TT,IPv4_NL,IPv4_TC,IPv4_TT"

while [ 1 ]; do

          LIST=`cat url.list`

          for i in $LIST; do

                    # echo -n "$i -> IPv4"

                    # curl -4 -w
'\nipv4_time_namelookup:%{time_namelookup}\nipv4_time_connect:%{time_co
nnect}\nipv4_http_code:%{http_code}\nipv4_time_total:%{time_total}\n' -
o /dev/null -s $i

                    # curl -4 -w '\nipv4_time_total:%{time_total}\n' -o
/dev/null -s $i

                    IPv4_NL=`curl -4 -w '%{time_namelookup}' -o
/dev/null -s $i`

                    IPv4_TC=`curl -4 -w '%{time_connect}' -o /dev/null -
s $i`

                    IPv4_TT=`curl -4 -w '%{time_total}' -o /dev/null -s
$i`

                    # echo -n "$i -> IPv6"

                    # curl -6 -w
'\nipv6_time_namelookup:%{time_namelookup}\nipv6_time_connect:%{time_co
nnect}\nipv6_http_code:%{http_code}\nipv6_time_total:%{time_total}\n\n'
-o /dev/null -s $i

                    # curl -6 -w '\nipv6_time_total:%{time_total}\n\n' -
o /dev/null -s $i

                    IPv6_NL=`curl -6 -w '%{time_namelookup}' -o
/dev/null -s $i`

                    IPv6_TC=`curl -6 -w '%{time_connect}' -o /dev/null -
s $i`

                    IPv6_TT=`curl -6 -w '%{time_total}' -o /dev/null -s
$i`

                    echo "$i, $IPv6_NL, $IPv6_TC, $IPv6_TT, $IPv4_NL,
$IPv4_TC, $IPv4_TT"

          done

sleep 60;

done
```

Sample output

The following is sample output (csv format) from the script above:

```
# FQDN,IPv6_NL,IPv6_TC,IPv6_TT,IPv4_NL,IPv4_TC,IPv4_TT
www.google.com, 0.004, 0.024, 0.045, 0.004, 0.028, 0.094
www.facebook.com, 0.028, 0.053, 0.107, 0.004, 0.053, 0.142
www.arin.net, 0.125, 0.069, 0.134, 0.067, 0.070, 0.134
www.google.com, 0.004, 0.024, 0.044, 0.004, 0.028, 0.099
www.facebook.com, 0.012, 0.053, 0.109, 0.004, 0.054, 0.140
www.arin.net, 0.012, 0.071, 0.135, 0.253, 0.070, 0.136
www.google.com, 0.004, 0.024, 0.046, 0.004, 0.028, 0.093
www.facebook.com, 0.060, 0.053, 0.107, 0.012, 0.055, 0.144
www.arin.net, 0.125, 0.069, 0.134, 0.125, 0.069, 0.136
```