Delay Metrology

Context

To assess the delay in a deployed network, one may easily exploit the results given by the ICMP ping command. However, this measure may not reflect the actual delay that applications may face. To properly measure the delay in a given infrastructure, we recommend to compare the results of various delay measurements jobs. Depending on the targeted evaluation (application level, transport level, network level, etc.), one measure may be more relevant than one other.

The most common way to assess delay in a deployed network remains to use round-trip time metric. Our choice is to use two tools one at ICMP level (fping) and one at UDP level (hping) to enforce the measurement.

However, this value may not reflect the real situation of the network of interest if the links from test host and to test host are unbalanced. To obtain this unbalanced delay assessment and better understanding of the tested network, one may use one-way delay tool which provides delay values on each link. The tool owping (composed of two jobs: owamp-client and owamp-server) is provided by OpenBACH in another scenario.

RTT Delay diagnostic including post processing comparison scenario

This scenario provides a delay diagnostic of a deployed network. It is assumed that the network is stable. The diagnostic allows to sequentially assess different delay metrics (RTT and one-way delay)

Objective

The purpose of this scenario is to compare delay results obtained from jobs: fping, hping (RTT) and owping (One-way Delay).

- `fping` measures the RTT delay (in ms) using the ICMP protocol (i.e. time since ICMP request sent until ICMP response received)
- `hping` measures the RTT delay (in ms) using the negotiation packets for opening a TCP connection (i.e. time since SYN sent until SYN-ACK received).
- `owping` measures the one-way delay (in ms) using OWAMP (One-way Active Measurement Protocol) detailed in RFC 4656. OWAMP requires a client/server architecture. A server daemon is installed on the test host and a TCP control connection is opened between the two client/server machines. Then, client launches owping command which sends test UDP packets. There are two jobs to correctly execute owping test: `owamp-client` and `owamp-server`. owamp-client returns two statistics: **owd_sent** (OWD from client to server) and **owd_received** (OWD from server to client). This tool has been implemented by perfSONAR.

Hping and fping (which are persistent jobs) are launched during 60s. Owamp-client is not a persistent job, it sends 100 test packets by default and after that the job instance stops. As it is shown in the

following scenarios, owamp-server job must be launched before owamp-client with a guard period to let enough time to the server to be switched on. This guard time depends on the current delay of the network of interest. In this scenario example, the guard time has been set to 5s.

Methodology

Two approaches are used to analyse the delay:

- The first one considers a parallel/simultaneous launch of the three jobs.
- The second one considers a sequential launch of the three jobs in order to avoid any overload caused by the injected test packets (active measurement methods).

The scenario proposed herein will then:

- Launch subscenarios delay_simultaneous or delay_sequential as subscenarios (allowing to compare the RTT measurement of fping, hping and owamp jobs).
- Launch two postprocessing jobs to compare the time-series and the CDF of the delay measurements.

These scenarios have been written using the scenario builder.

How to launch it

The scenario is available in network_delay . It uses helpers (see API scenario manual for more information on helpers), and subscenarios `delay_sequential` and `delay_simultaneous`.

You must already have a project (i.e. “your_project”), two entities in the project (a server and a client), the hping/fping/owamp-client jobs installed in the client and the owamp-server job installed in the server. You also need to install histogram and time-series jobs on the “your_entity”.

The reference scenario script generate_network_delay will allow to launch it as follows.

```
python3 generate_network_delay.py -o -p your_project --client your_client_entity --server
your_server_entity --ip_dst @IP --entity_pp your_entity run
```

By default, it launches the scenario with the simultaneous methodology. If you want to launch the sequential one, you should add “- -sequential” to your arguments.

Alternatively, it can create a JSON file that could be imported to the OpenBACH web interface:

```
python3 generate_network_delay.py -o -p your_project --client your_client_entity --server
your_server_entity --ip_dst @IP --entity_pp your_entity build .
```

An example of the type of results that this scenarios is capable of plotting is shown below:

The plots can be downloaded from the OpenBACH web interface: go to the scenario instance, click on export CSV, check the histogram and time-series boxes and click on Download.