Yaskawa solectria

mesh network performance test tool

Features, screen captures & results

mesh network performance test tool

Nov 20, 2016

Features, screen captures & results

# Objective

Testing the connectivity of each and every node in mesh network is necessary to ensure desired network performance. In addition, the quality of the links is also to be measured using simple tool that does not require the knowledge of sophisticated languages and commands. Considering the limited memory available in devices, the performance tool should be able to run without calling for additional software and packages.

The objective is to build a light weight network performance test tool that checks for:

* List of neighbors
* Connectivity to each neighbor
* Latency from test node to each neighbor (in ms)
* Percentage of data loss from test node to each neighbor
* Upload throughput (Mbps)
* Download throughput (Mbps)
* Next hop (in future)

The tool uses the tools that are already available in kernel.

# Tool architecture & features

The performance tool is designed to use simple kernel utilities such as ping and iperf that are already available in the kernel. The tool can be invoked by a single command either through the command line interface or by clicking on a button in the web GUI as shown in Figure 1 (to be designed in future).

Assumptions made in building the tool:

1. iperf server daemon is running in each node in default port

2 network\_map.json file has IPv4 entries

3. All the nodes wake up in the same channel

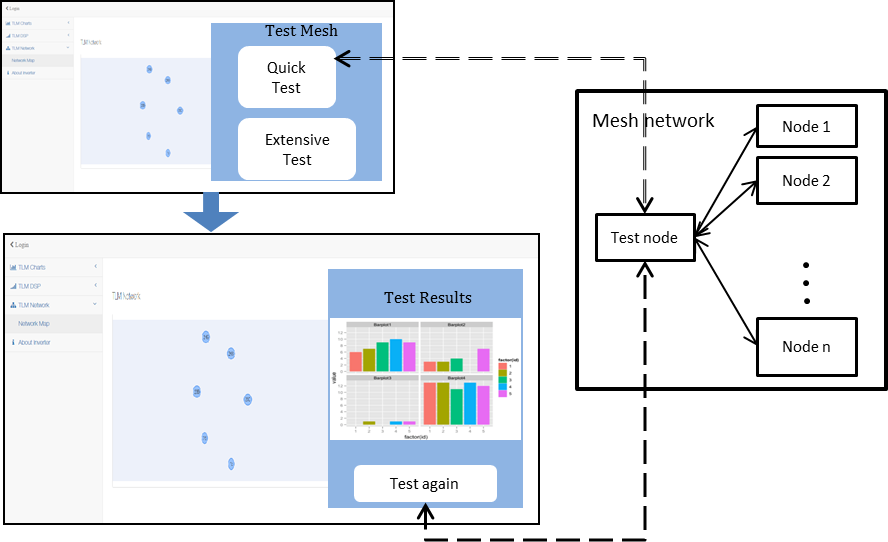


Figure 1. Performance tool model and layout

The performance testing tool script can be invoked from any device in mesh. Upon invocation, the tool does the following:

1. Create the documents and folders required for the execution
2. An iperf server deamon is waiting in every terminal. This instance is to be suspended before running the bidirectional iperf command.
3. Identify the neighbors from /etc/solectria/network/network\_map.json file
4. Fetch IP address, stored as the 2nd field in each row
5. Ping each neighboring terminal and estimate the loss percentage
6. Estimate the average latency in milliseconds
7. The upload and download throughput are determined using iperf
   1. One terminal (probably the gateway) acts as the iperf client and connects with each iperf server running in each device to measure the upload throughput
   2. Another set of connections are made by reversing the server-client roles to determine the download throughput
8. The results are displayed in table format in command line and also stored as key-value pairs in Nw\_perf.json file

The script handles the following corner cases:

* Longer ping delays
* Longer iperf delays
* Link connectivity issues
* Nodes becoming unreachable during test

There is a function to restrict the span of each command that can be extended in future with desired duration of each command.

**Modes of running:**

1. Quick execution mode: In this mode, the connectivity and link quality can be quickly checked by skipping the throughput computation. On average, this mode takes around 7 secs for each node.

2. Extensive execution mode: In this mode, in addition to the connectivity and link quality the throughput is also computed. The average time spent for each node in this mode is approximately 25 secs. Thus the total time linearly increases with the network size.

The options are displayed along with the estimated total worst case computing time to the user. The unexpected user inputs are handled by carefully terminating the execution.

**Results:**

The results are displayed in table format in the command line interface and also saved as Nw\_perf.json file. The .json file can be pulled from a web interface to generate plots and tables. The log files generated during execution are cleaned up after execution. Only results in Nw\_perf.json file remains!

# Test procedure

A 5 node (Veriscite boards) mesh network is set-up indoors in office. The tool is run on a test node have one server (S: 10.15.140.239 – Vericite board) and 4 clients:

C1: 10.15.141.76 (Vericite board)

C2: 10.15.141.82 (Vericite board)

C3: 10.15.140.246 (Vericite board)

C3: 10.15.140.248 (Vericite board)

Nodes C1 and C2 are placed closer to S (as shown in Figure.1), while C3 was separated by around 35 feet from S and C4 is placed at around 50 feet away from server.

S

C2

C1 (Scenario 1)

C4 (Scenario 2)

C3

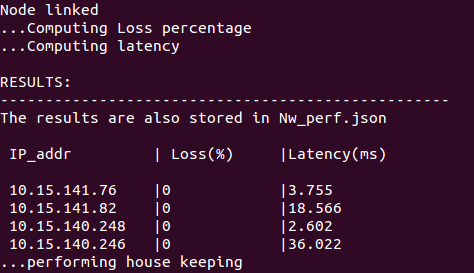
# test results

The script execution screen shots is given below:

1. Displaying the execution time for each mode and getting user input:

## 

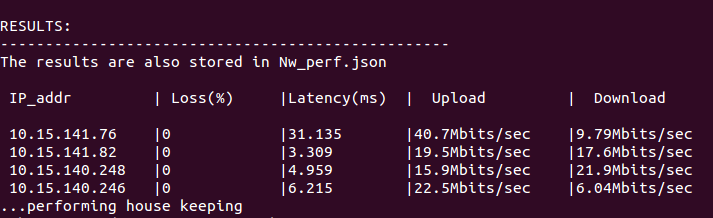
1. Checking for node connectivity and computing only loss percentage and latency. Also displaying the results in table format.



1. In mode 2, checking for node connectivity and computing loss percentage, latency and throughputs.

## 

1. The results are displayed in table format and housekeeping is performed after execution.



1. The results are stored in .json format for mode 1 and mode 2:

## 

## 