OpenThread Quality Dashboard

Provided by OpenThread

Version 1.0

Table of Contents

Revision History

Date	**Version**	**Descriptions**
Feb 12, 2019	Version 1.0	Initial version. Introduction of OpenThread Quality Dashboard

1. Purpose

OpenThread Quality Dashboard, owned and maintained by the Google OpenThread Team, shows the quality of OpenThread iteratively along with the development on GitHub. OT Quality Dashboard comprises two sub-Dashboards: Certification and Performance.

The conformance with the Thread Specification and the interoperability with other Thread stack vendors are evaluated by the Thread Test Harness and are presented in Certification Dashboard. Please refer to https://openthread.io/certification for more information about certification.

OpenThread Performance Dashboard (“OT Perf Dashboard”) illustrates the performance metrics of OpenThread as a wireless networking protocol implementation. This document serves as a supplement and introduces:

- Test methodology for each performance metric
- How to interpret the OT Perf Dashboard
- Performance summary under an OpenThread reference release commit

The results of reference release commit 55bf9fc2 will be taken as a benchmark for further regular regression.

This document will be updated as new metrics and scenarios are added to OT Perf Dashboard. Refer to https://openthread.io/testing/quality-dashboard for the most up-to-date and detailed performance/certification results.

2. Overview

OpenThread performance is evaluated across a variety of metrics. The test environment, methodology, and summary of the results are covered under the following metrics and use cases:

- Metrics
    - Latency
    - Loss Rate
    - Throughput
- Topology
    - 12-node, 2-cluster, conductive with attenuators

Ideally, performance should be evaluated across a number of different use cases and deployment scenarios. This is an initial step to demonstrating the OT Perf Dashboard platform. Over time, additional use cases and network topologies will be included in the OT Quality Dashboard.

3. Topology and Testbed

The system under test is composed of 12 nodes, which form a 2-cluster topology. The nodes in each cluster are connected completely, and the two clusters are connected together by only one link. All nodes are Routers (including the Leader) in the Thread network. The performance metrics introduced in Sections 4 - 6 are all tested using this topology, shown in Figure 3.1.

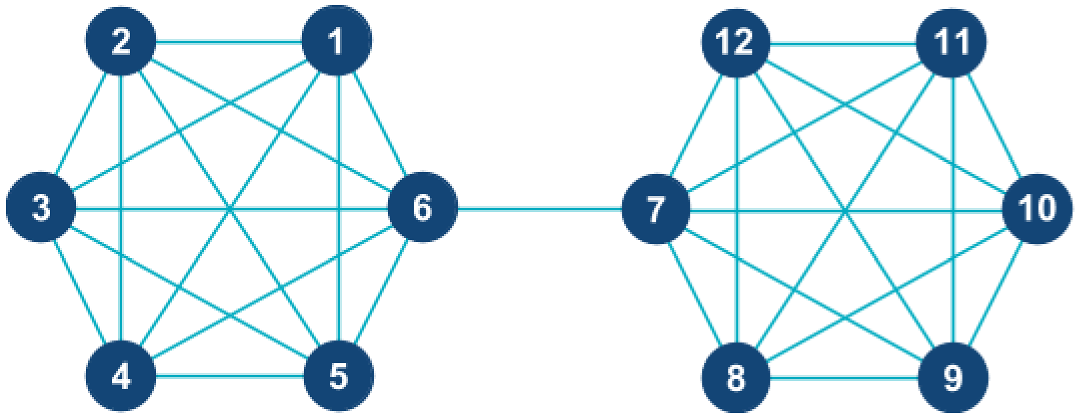


Figure 3.1 Topology of 12-node, 2-cluster

The devices compliant with OpenThread ("OT device") are deployed as the topology in the testbed. The deployment of OT devices connected with cables, attenuators, and splitters is illustrated in Figure 3.2. More specifically, all the results delivered in the OT Perf Dashboard are using nRF52840 PDKs as the OT devices in the testbed. The testbed is not limited to a specific platform, correspondingly it applies to all kinds of OT devices.

In this testbed, one GPIO pin from each OT Device is connected to a 1-wire bus. The performance metrics introduced in Sections 4 - 6 are all tested using this testbed. The usages of the testbed for each use case are different, as detailed in the procedures.

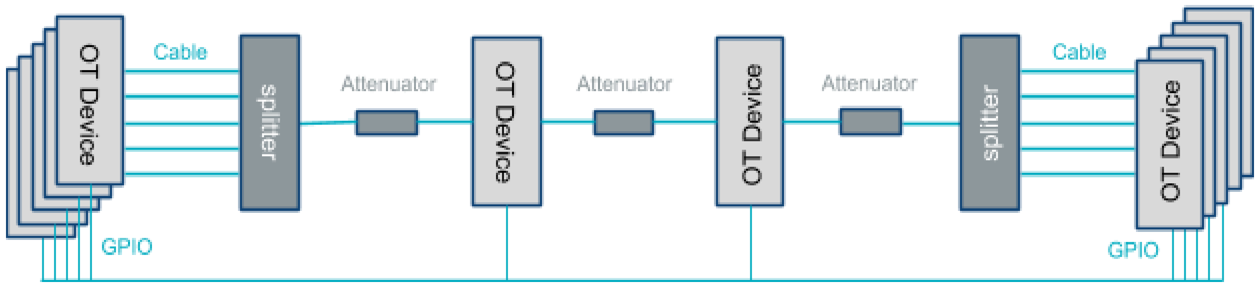


Figure 3.2 Testbed deployment

4. Latency

Latency tests measure and analyze latency metrics over 1, 2, and 3 hop(s) with different UDP payload sizes.

4.1. Methodology and Procedure

In order to get accurate latency, GPIOs are used to synchronize events across the testbed. A GPIO interrupt, which is triggered at the packet source, will reach all the devices at the same time and trigger the devices in RX state to record the packet sent time. Latency is then measured by computing the difference between received and sent timestamps. An example GPIO interrupt time sequence is shown in Figure 4.1. Each UDP packet is tagged with an identifying sequence ID to make sure the measurement is correct.

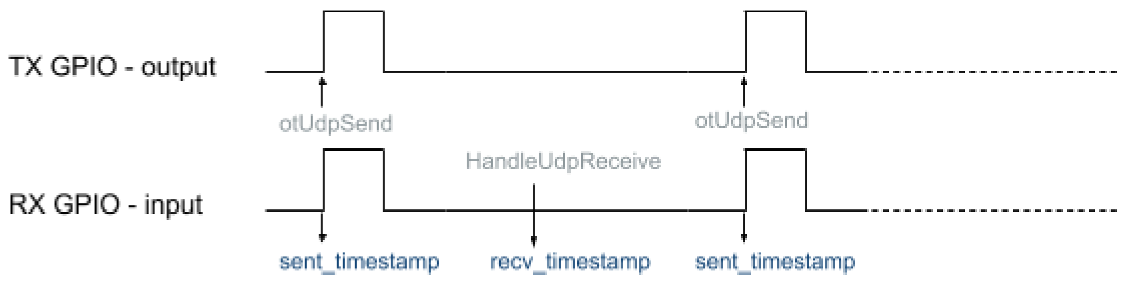


Figure 4.1 Accurate latency time sequence

Unicast Latency tests are performed for each pair of nodes. Detailed procedures of the tests are listed in Table 4.1.

Table 4.1 Unicast latency test steps

Step	Device	Description
1	All	Connect GPIO pins of all Nordic dev boards together using dupont lines.
2	All	Verify topology is formed correctly.
3	Device 1	Select Device 1 as the source.
4	Device 2	Select Device 2 as the destination.
5	Device 1	Send a stream of frames with the P (P = 64) bytes UDP payload size.

6	Device 1	Output a GPIO signal when sending the frame.
7	Device 2	Record the timestamp A when the GPIO interrupt triggered.
8	Device 2	Receive the UDP packet with the identifying tag from the Device 1 and record the timestamp B.
9	Device 2	Calculate the one-way latency by (B-A).
10	Device 1 , Device 2	Repeat above Step 5 ~ 9 with T ms (T = 500) for N times (N = 50)
11	Device 1, Device 2	Repeat Step 5 to Step 10 with different UDP payload size PN(PN = 128, 256, 512) bytes.
12	Device 1 , Device X (X = 3, 4, …, 12)	Repeat Step 5 to Step 11 with Device X as the destination, respectively.
13	Device X (X = 2, 3, …, 12), Device Y (Y = 1, 2, …, 12 and Y != X)	Repeat Step 4 to Step 12 with Devices X as the source, and Y as the destination respectively.
14	All	Collect statistics (Min, Max, Avg, mode, median, 90%*) of latency over 1, 2, and 3 hop links with different payload sizes. (Note *: At least 90% packets are received within * ms).

4.2. Results

Latency test results are analyzed by number of hops, payload size, historical commits, etc. The summary of latency test results is found on the main Performance Dashboard page. The following sections will give typical examples on how to interpret the visualized results and the overview of the reference release commit.

4.2.1. Examples

A Cumulative Distribution Function (CDF) is used to show the latency distribution of all received packets, revealing the summary and a general picture of the latency performance. The latest commit is highlighted in red, compared with historical results (reference release commit 55bf9fc2). The differences between the latest commit and the baseline can be seen at first glance. It is helpful to visualize any change in performance of different hops and payload size.

Take CDF for Unicast Latency of a 64-byte payload in Figure 4.2 as an example. We can see that:

- The latest latency curve nearly overlaps with the release commit, which illustrates minimal change in performance.
- The latency range of a 64-byte payload size for 1 hop is approximately 5~8 ms.

Precise metric numbers can be found in the following paragraphs.

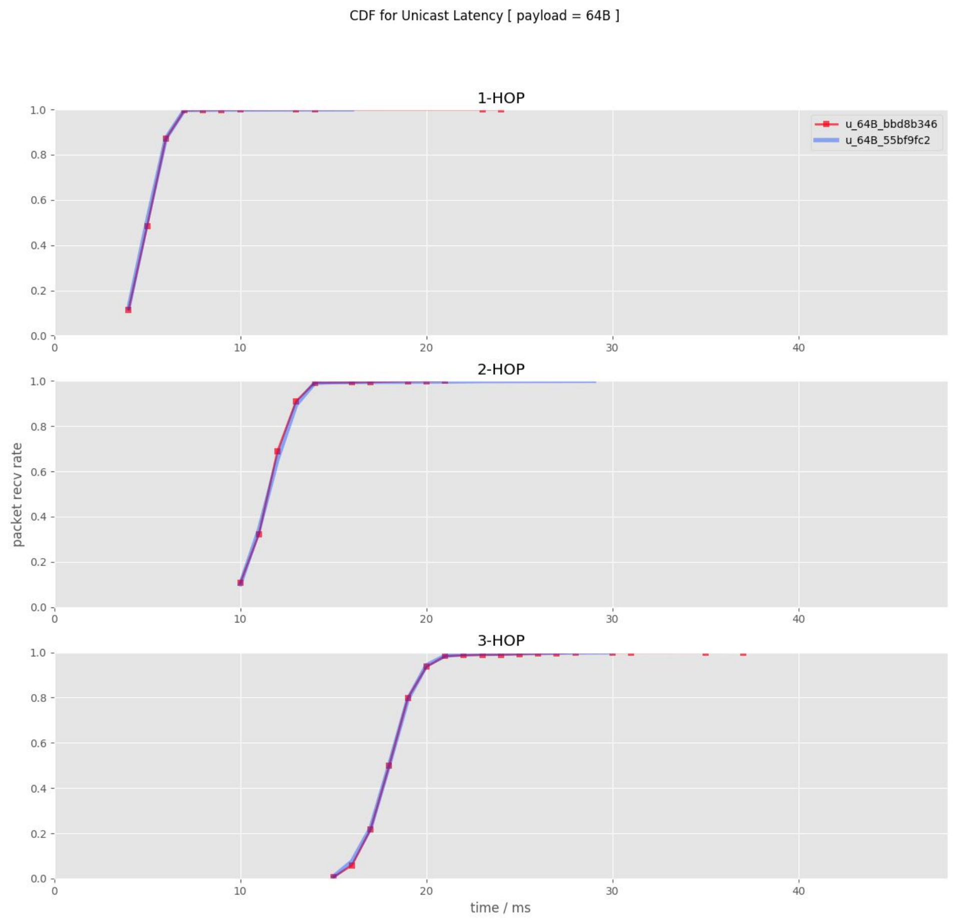


Figure 4.2 CDF for unicast latency [64-byte payload size]

The detailed latency results could be found on the Average Latency page of OT Perf Dashboard. Average unicast latency (ms) of different commits is shown in Figure 4.3.

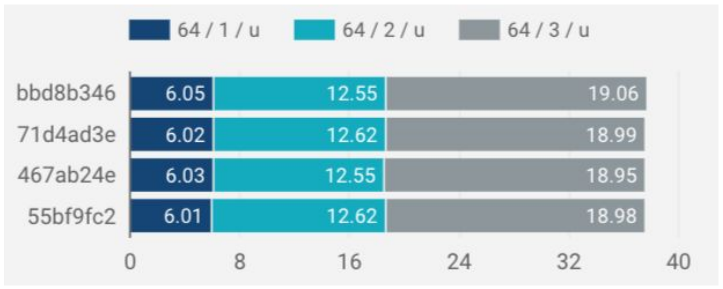


Figure 4.3 Average unicast latency [64-byte payload size]

The CDF for unicast latency of each node is shown in Figure 4.4, which helps to identify potential problems in this run by providing results of different nodes as source and destination respectively.

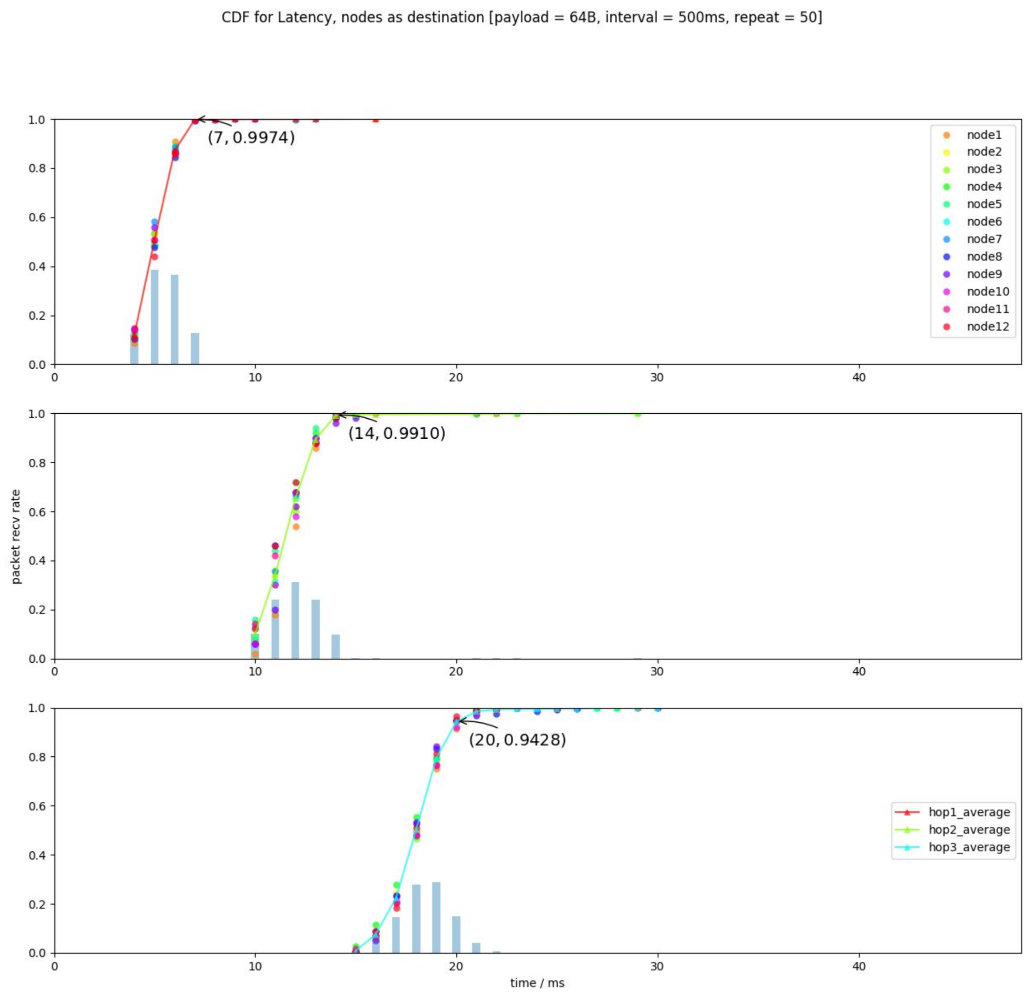


Figure 4.4 CDF for unicast latency of each node [64-byte payload size]

In Table 4.2, latency detail analysis includes the following statistics value for each hop: average, median, mode, minimum, maximum, and 90th percentile, which means at least 90% packets are received within a certain time period.

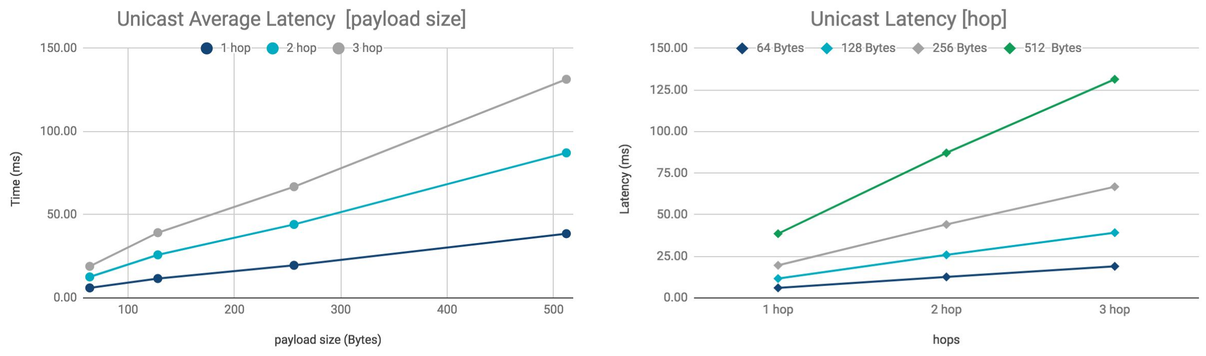
Table 4.2 Unicast latency (ms) of *commit* [55bf9fc2] [64-byte payload size]

Hop	Average	Median	Mode	Min	Max	90%*
1	6.01	5.92	5.2	4.852	16.967	7
2	12.62	12.573	13.2	10.315	29.816	14
3	18.98	18.982	19	15.534	30.06	20

4.2.2. Overview of Reference Release Commit

The latency test overview of different payload size and hops for the reference release commit (55bf9fc2) are shown in Figure 4.5. All the results are summarized from data on the OT Perf Dashboard, from which we can see:

- For a payload size of 64 bytes for 3 hops, latency is maintained at less than 20 ms. This case is the most common application where the payload is without fragmentation and hops are within 3 hops.
- For 512 bytes, 1 hop, the latency is under 40ms; up to 3 hops for 512 bytes, latency is around 130 ms which is less than the 200 ms often desired for human interaction with the devices.



(a) (b)

Figure 4.5 Average unicast latency varying with (a) payload size, (b) hops

5. Loss Rate

5.1. Methodology and Procedure

Packet loss is measured at the same time during the latency tests. Destination devices will not record the received timestamp if a packet has been lost. The loss rate is calculated over 1, 2, and 3 hop(s) with different UDP payload sizes. Packets loss rate is tested using the same topology with latency test as well, refer to Figure 3.1.

5.2. Results

The summary of loss rate result is found on the main Performance Dashboard page. The following sections will give typical examples on how to interpret the visualized results and the overview of the reference release commit.

5.2.1. Examples

The CDF figure (Figure 4.2) presents an overview: The loss rate of latency packet is 0 as the cumulative distribution reaches 100%.

Packet Loss Rate for every connection is shown in detail in Figure 5.1. All the packets are received without packet loss between each pair of nodes, which is consistent with Figure 4.2.

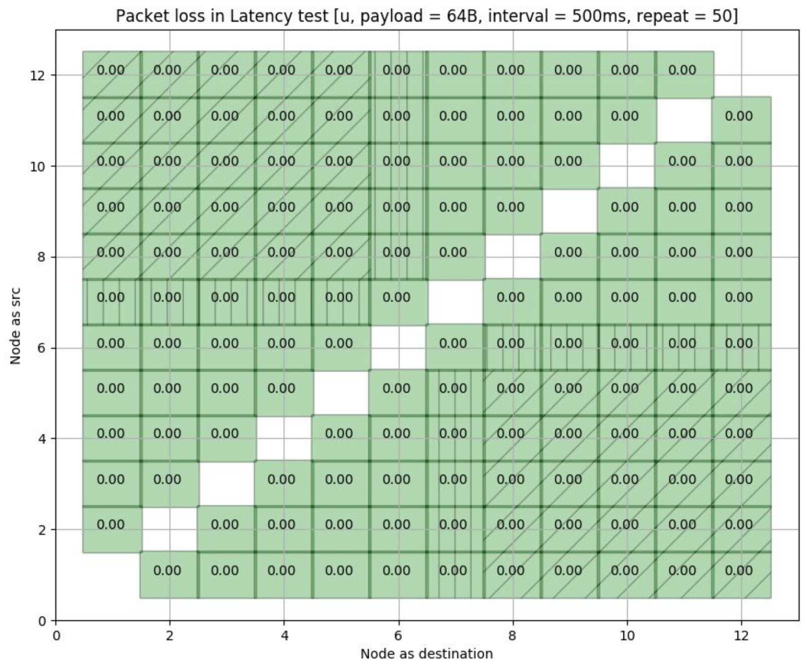


Figure 5.1 Packet loss rate of unicast latency [64-byte payload size]

The boxes of packet loss rate are marked with green / yellow / red to illustrate health / warning / error:

- < green >: lose rate < 0.1
- < yellow >: 0.1 =< lose rate < 0.2
- < red >: lose rate >= 0.2

The boxes are also marked with hatching to illustrate hops:

- < null > = 1 hop
- < ' | ' > = 2 hops
- < ' / ' > = 3 hops

5.2.2. Overview of Reference Release Commit

The loss rate overview for the reference release commit (55bf9fc2) is shown in Table 5.1. All the results are summarized per data on the OT Perf Dashboard, from which we could see:

- Packet loss rate is 0% under typical UDP payload size: 64, 128, 256, and 512 bytes for hops 1~3.

Table 5.1 Summary of Packet loss rate

Hop	Packet loss rate of different payload size (Bytes)			
	64	128	256	512
1	0%	0%	0%	0%
2	0%	0%	0%	0%
3	0%	0%	0%	0%

6. Throughput

6.1. Methodology and procedures

Throughput is tested by sending UDP packets of different payload sizes using the testbed shown in Figure 3.2. The source will send UDP packets at the target bandwidth. The target bandwidth increases until the loss rate goes beyond a certain threshold at the destination. An example of throughput traffic flow is shown in Figure 6.1.

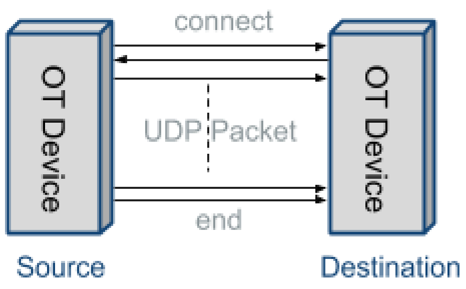


Figure 6.1 Traffic flow

Throughput tests are performed for every possible connection between any two nodes. Detailed procedures of the tests are listed in Table 6.1.

Table 6.1 Throughput test steps

Step	Device	Description
1	All	Verify topology is formed correctly.

2	Device 2	Set Device 2 as the source.
3	Device 1	Set Device 1 as the destination.
4	Device 2	Send frames with the N (N = 64) bytes payload size UDP packet at A (A = 10) kbps to the destination for a period of time T seconds (T = 11). Packets are sent at an even interval.
5	Device 1	Count the frames that are received from the source. Get the packet loss ratio L, $L = (N_{\text{sent frames}} - N_{\text{received frames}}) / N_{\text{sent frames}}$ If L <= P (P = 10%): Go to Step 6; else: Go to Step 7.
6	Device 2	$B_i = A + S$ (S <= 20kbps, adaptive according to payload size and hops) Send frames lasting for T time with the N bytes UDP payload size at the expected bandwidth B_i kbps; $A = B_i$ Go to Step 5.
7	Device 1, 2	Stop sending packets and obtain the value of A, the 1 hop throughput is A * (1 - L)
8	Device 1, 2	Repeat Step 5 to Step 7 with different UDP payload size PN (PN = 128, 256, 512) bytes.
9	Device 1 , Device X (X = 3, 4, …, 12)	Select Device X as the source, repeat Step 3 to Step 8 to measure the throughput.
10	Device X (X = 2, 3, …, 12), Device Y (Y = 1, 2, …, 12 and Y != X)	Repeat Step 4 to Step 9 with Devices X as the source, and Y as the destination respectively.

6.2. Results

The summary of throughput result is found on the main Performance Dashboard page. The following sections will give typical examples on how to interpret the visualized results and the overview of reference release commit.

6.2.1. Examples

Throughput test results are analyzed by hops, payload size, historical commits.

The throughput summary figure plots the achieved throughput relative to the data rate. The throughput of the latest commit is highlighted in red to compare with historical results in blue (reference release commit 55bf9fc2). This figure provides a general picture of throughput performance, with differences between the latest commit and the baseline easily discernible, and provides a visual comparison of performance for different hops and payload size.

Take the throughput summary of a 64-byte payload, as shown in Figure 6.2. The plot shows the achieved throughput relative to data rate. The latest throughput curve nearly overlaps with the release commit, which illustrates minimal change in the throughput results.

From this figure we can see that:

- The throughput value of a 64-byte payload size for 1 hop is 90 kbps.
- The throughput has not decreased even though the sent bandwidth greatly exceeded the received bandwidth.

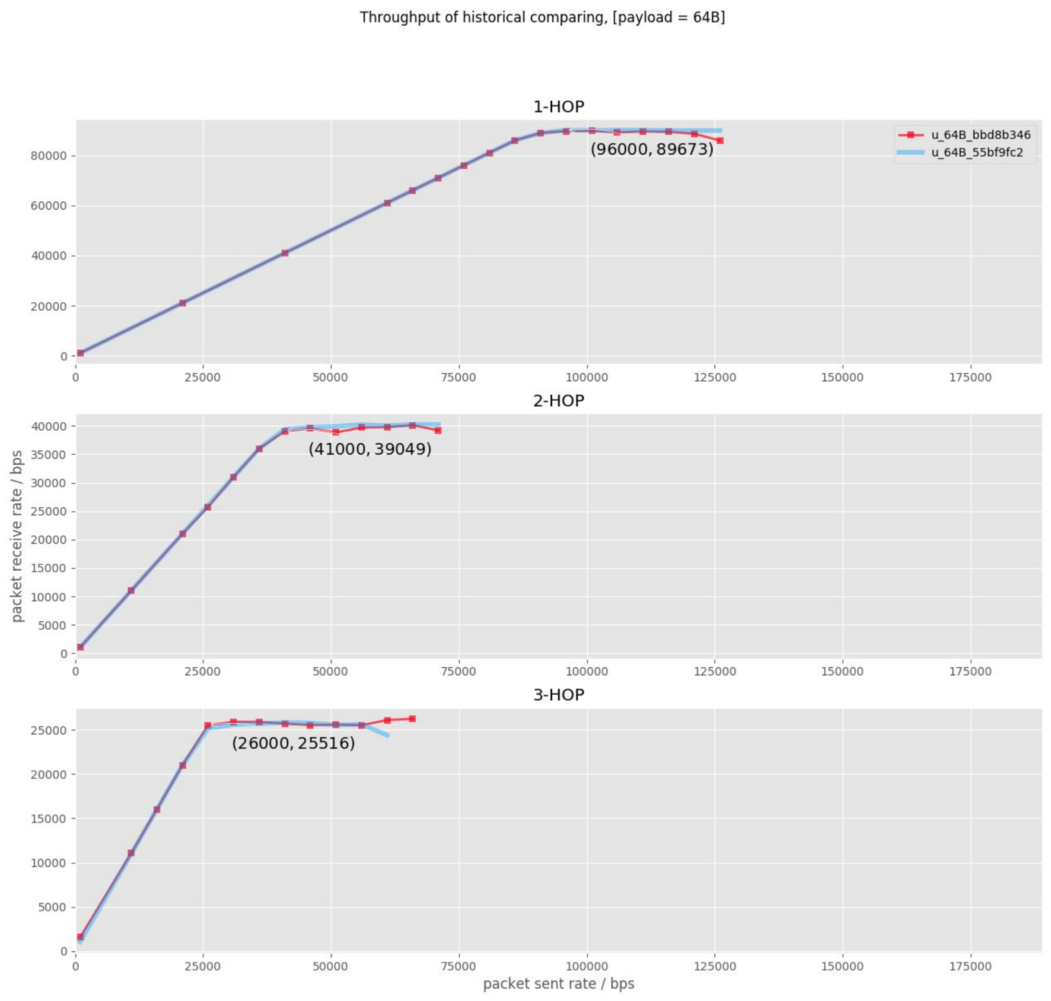


Figure 6.2 Throughput summary: comparison of latest and historical commits [64-byte payload size]

6.2.2. Overview of Reference Release Commit

The throughput overview for the reference release commit (55bf9fc2) are shown in Figure 6.3. All the results are summarized from data on the OT Perf Dashboard, from which we see:

- For the payload size of 64, 128, 256, 512 bytes, throughput results for 1 hop are in range of 90 kbps ~ 108 kbps.
- Throughput and latency results are consistent under mutual conversion for each payload size and hop.

Note: Use formula "Throughput_conversion = 1000 /Latency(ms) * Payload Size (Bytes) * 8". Take 512B, 1 hop as an example. Refer to latency results in Figure 4.5, Latency = 38.6 ms, Throughput_conversion = 106.1 bps. Throughput_conversion is consistent with the real output Throughput.

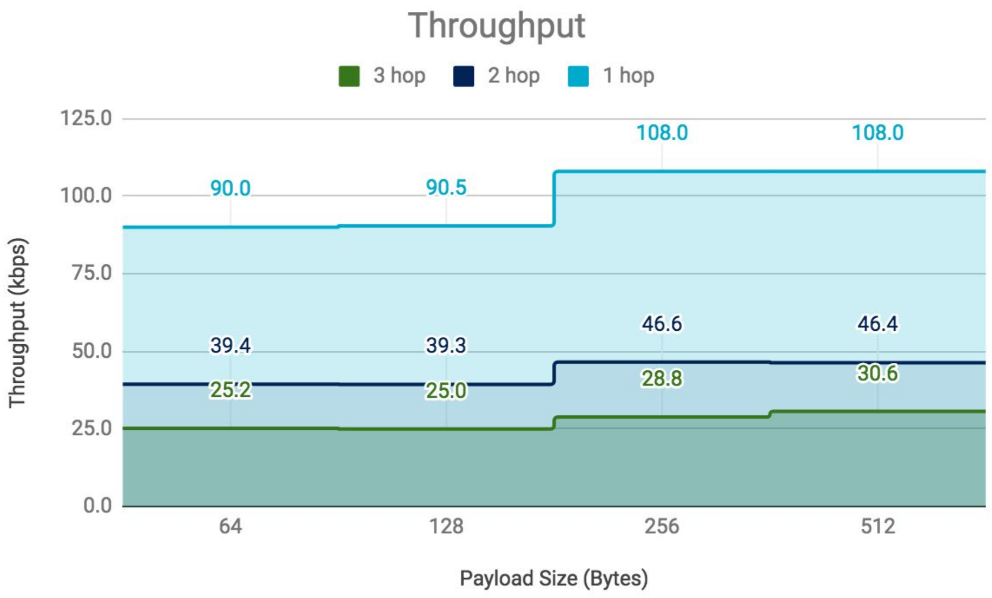


Figure 6.3 Throughput under different payload size and hops

7. Summary

- OT Perf Dashboard provides latency, loss rate, and throughput metrics in a typical topology for various payload sizes and hops.
- OT Quality Dashboard will be updated periodically and continuously based on the OpenThread master branch.