CSIT REPORT

Release rls1801_2

Jun 01, 2018

Contents

CHAPTER 1

Introduction

1.1 Overview

This is the **F**ast **D**ata **I/O** Project (**FD.io**) **C**ontinuous **S**ystem **I**ntegration and **T**esting (**CSIT**) project report for CSIT rls1801_2 system testing of VPP-18.01.2 release.

This is the full html version, there is also a reduced PDF version of this report[1].

The report describes CSIT functional and performance tests and their continuous execution delivered in CSIT rls1801_2. A high-level overview is provided for each CSIT test environment running in LF (Linux Foundation) FD.io Continuous Performance Labs. This is followed by summary of all executed tests against the VPP-18.01.2 release and associated FD.io projects and sub-systems (Honeycomb, DPDK, NSH_SFC), CSIT rls1801_2 release notes, result highlights and known issues discovered in CSIT. More detailed description of each environment, pointers to CSIT test code documentation and detailed test resuls with links to the source data files are also provided.

CSIT rls1801_2 report contains following main sections and sub-sections:

1. **Introduction** - general introduction to CSIT project; *Overview* - this section; *CSIT Test Naming* - CSIT general naming convention for test suites and test cases, important to recognize the high-level test content and interpret reported results; *General Notes* - general notes related to this report.
2. **VPP Performance Tests** - VPP performance tests executed in physical FD.io testbeds; *Overview* - tested topologies, test coverage and naming specifics, methodology for multi-core, packet throughput and latency, and KVM VM vhost tests; *CSIT Release Notes* - changes in CSIT rls1801_2, added tests, performance changes, environment or methodology changes, known CSIT issues; *Packet Throughput Graphs* and *Packet Latency Graphs* - plotted NDR, PDR throughput and latency results from multiple test job executions; *Test Environment* - environment description; *Documentation* - CSIT source code documentation for VPP performance tests.
3. **VPP Functional Tests** - VPP functional tests executed in virtual FD.io testbeds; *Overview* - tested virtual topologies, test coverage and naming specifics; *CSIT Release Notes* - changes in CSIT rls1801_2, added tests, environment or methodology changes, known CSIT issues, tests to be added; *Test Environment* - environment description ; *Documentation* - source code documentation for VPP functional tests.
4. **Honeycomb Functional Tests** - Honeycomb functional tests executed in virtual FD.io testbeds; *Overview* - tested virtual topologies, test coverage and naming specifics; *CSIT Release Notes* - changes

[1] https://docs.fd.io/csit/rls1801_2/report/_static/archive/csit_rls1801_2.pdf

in CSIT rls1801_2, added tests, environment or methodology changes, known CSIT issues; *Test Environment* - environment description ; *Documentation* - source code documentation for Honeycomb functional tests.

5. **NSH_SFC Functional Tests** - NSH_SFC functional tests executed in virtual FD.io testbeds; *Overview* - tested virtual topologies, test coverage and naming specifics; *CSIT Release Notes* - changes in CSIT rls1801_2, added tests, environment or methodology changes, known CSIT issues; *Test Environment* - environment description ; *Documentation* - source code documentation for NSH_SFC functional tests.
6. **Detailed Test Results** - auto-generated results from CSIT jobs executions using CSIT Robot Framework output files as source data; *VPP Performance Results*, *VPP Functional Results*, *Honeycomb Functional Results*, *VPPtest Functional Results*.
7. **Test Configuration** - auto-generated DUT configuration data from CSIT jobs executions using CSIT Robot Framework output files as source data; *VPP Performance Test Configs*, *VPP Functional Test Configs*.
8. **Test Operational Data** - auto-generated DUT operational data from CSIT jobs executions using CSIT Robot Framework output files as source data; *VPP Performance Operational Data*.
9. **CSIT Framework Documentation** - description of the overall CSIT framework design hierarchy, CSIT test naming convention, followed by description of Presentation and Analytics Layer (PAL) introduced in CSIT rls1801_2.

1.2 General Notes

All CSIT test results listed in this report are sourced and auto-generated from `output.xml` RF (Robot Framework) files resulting from LF FD.io Jenkins jobs execution against VPP-18.01.2 release release artifacts. References are provided to the original LF FD.io Jenkins job results. However, as LF FD.io Jenkins infrastructure does not automatically archive all jobs (history record is provided for the last 30 days or 40 jobs only), additional references are provided to the RF result files that got archived in FD.io nexus online storage system.

FD.io CSIT project currently covers multiple FD.io system and sub-system testing areas and this is reflected in this report, where each testing area is listed separately, as follows:

1. **VPP - Performance** - VPP benchmarking tests are executed in physical FD.io testbeds, focusing on VPP network data plane performance at this stage, both for Phy-to-Phy (NIC-to-NIC) and Phy-to-VM-to-Phy (NIC-to-VM-to-NIC) forwarding topologies. Tested across a range of NICs, 10GE and 40GE interfaces, range of multi-thread and multi-core configurations. VPP application runs in host user-mode. TRex is used as a traffic generator.
2. **LXC and Docker Containers VPP memif - Performance** - VPP memif virtual interface tests interconnect multiple VPP instances running in containers. VPP vswitch instance runs in bare-metal user-mode handling Intel x520 NIC 10GbE interfaces and connecting over memif (Master side) virtual interfaces to more instances of VPP running in LXC or in Docker Containers, both with memif virtual interfaces (Slave side). Tested across a range of multi-thread and multi-core configurations. TRex is used as a traffic generator.
3. **Container Topologies Orchestrated by K8s - Performance** - CSIT Container topologies connected over the memif virtual interface (shared memory interface). For these tests VPP vswitch instance runs in a Docker Container handling Intel x520 NIC 10GbE interfaces and connecting over memif (Master side) virtual interfaces to more instances of VPP running in Docker Containers with memif virtual interfaces (Slave side). All containers are orchestrated by Kubernetes, with Ligato[2] for container networking. TRex is used as a traffic generator.
4. **VPP Functional** - VPP functional tests are executed in virtual FD.io testbeds focusing on VPP packet processing functionality, including network data plane and in -line control plane. Tests cover vNIC-to-vNIC vNIC-to-VM-to-vNIC forwarding topologies. Scapy is used as a traffic generator.

[2] https://github.com/ligato

5. **Honeycomb Functional** - Honeycomb functional tests are executed in virtual FD.io testbeds, focusing on Honeycomb management and programming functionality of VPP. Tests cover a range of CRUD operations executed against VPP.
6. **NSH_SFC Functional** - NSH_SFC functional tests are executed in virtual FD.io testbeds focusing on NSH_SFC of VPP. Tests cover a range of CRUD operations executed against VPP.

In addition to above, CSIT rls1801_2 report does also include VPP unit test results. VPP unit tests are developed within the FD.io VPP project and as they complement CSIT system functional tests, they are provided mainly as a reference and to provide a more complete view of automated testing executed against VPP-18.01.2 release.

FD.io CSIT system is developed using two main coding platforms RF and Python. CSIT rls1801_2 source code for the executed test suites is available in CSIT branch rls1801_2 in the directory `./tests/<name_of_the_test_suite>`. A local copy of CSIT source code can be obtained by cloning CSIT git repository - **`git clone https://gerrit.fd.io/r/csit`**. The CSIT testing virtual environment can be run on a local computer workstation (laptop, server) using Vagrant by following the instructions in CSIT tutorials[3].

[3] https://wiki.fd.io/view/CSIT#Tutorials

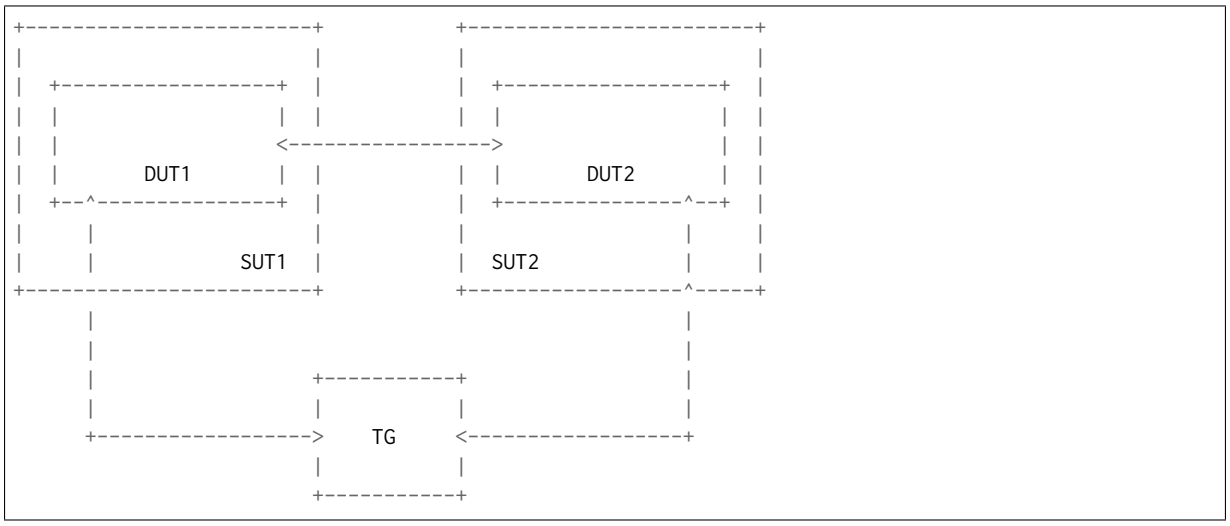
CHAPTER 2

VPP Performance Tests

2.1 Overview

2.1.1 Tested Physical Topologies

CSIT VPP performance tests are executed on physical baremetal servers hosted by LF FD.io project. Testbed physical topology is shown in the figure below.:



SUT1 and SUT2 are two System Under Test servers (Cisco UCS C240, each with two Intel XEON CPUs), TG is a Traffic Generator (TG, another Cisco UCS C240, with two Intel XEON CPUs). SUTs run VPP SW application in Linux user-mode as a Device Under Test (DUT). TG runs TRex SW application as a packet Traffic Generator. Physical connectivity between SUTs and to TG is provided using different NIC models that need to be tested for performance. Currently installed and tested NIC models include:

1. 2port10GE X520-DA2 Intel.
2. 2port10GE X710 Intel.
3. 2port10GE VIC1227 Cisco.
4. 2port40GE VIC1385 Cisco.

5. 2port40GE XL710 Intel.

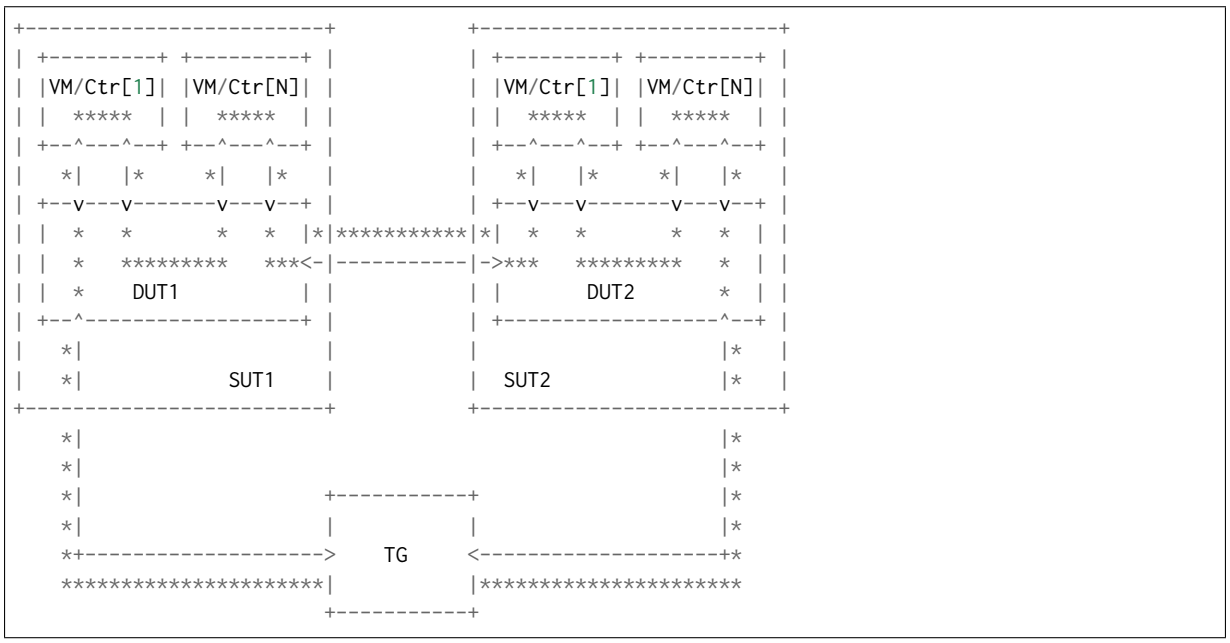
From SUT and DUT perspective, all performance tests involve forwarding packets between two physical Ethernet ports (10GE or 40GE). Due to the number of listed NIC models tested and available PCI slot capacity in SUT servers, in all of the above cases both physical ports are located on the same NIC. In some test cases this results in measured packet throughput being limited not by VPP DUT but by either the physical interface or the NIC capacity.

Going forward CSIT project will be looking to add more hardware into FD.io performance labs to address larger scale multi-interface and multi-NIC performance testing scenarios.

For service chain topology test cases that require DUT (VPP) to communicate with VirtualMachines (VMs) or with Linux/Docker Containers (Ctrs) over vhost-user/memif interfaces, N of VM/Ctr instances are created on SUT1 and SUT2. Three types of service chain topologies are tested in CSIT rls1801_2:

1. "Parallel" topology with packets flowing from NIC via DUT (VPP) to VM/Container and back to VPP and NIC;
2. "Chained" topology (a.k.a. "Snake") with packets flowing via DUT (VPP) to VM/Container, back to DUT, then to the next VM/Container, back to DUT and so on until the last VM/Container in a chain, then back to DUT and NIC;
3. "Horizontal" topology with packets flowing via DUT (VPP) to Container, then via "horizontal" memif to the next Container, and so on until the last Container, then back to DUT and NIC. "Horizontal" topology is not supported for VMs;

For each of the above topologies, DUT (VPP) is tested in a range of L2 or IPv4/IPv6 configurations depending on the test suite. A sample DUT "Chained" service topology with N of VM/Ctr instances is shown in the figure below. Packet flow thru the DUTs and VMs/Ctrs is marked with ***:



In above "Chained" topology, packets are switched by DUT multiple times: twice for a single VM/Ctr, three times for two VMs/Ctrs, N+1 times for N VMs/Ctrs. Hence the external throughput rates measured by TG and listed in this report must be multiplied by (N+1) to represent the actual DUT aggregate packet forwarding rate.

For a "Parallel" and "Horizontal" service topologies packets are always switched by DUT twice per service chain.

Note that reported DUT (VPP) performance results are specific to the SUTs tested. Current LF FD.io SUTs are based on Intel XEON E5-2699v3 2.3GHz CPUs. SUTs with other CPUs are likely to yield different results. A good rule of thumb, that can be applied to estimate VPP packet thoughput for Phy-to-Phy (NIC-to-NIC, PCI-to-PCI) topology, is to expect the forwarding performance to be proportional to CPU core frequency, assuming CPU is the only limiting factor and all other SUT parameters equivalent to FD.io CSIT

environment. The same rule of thumb can be also applied for Phy-to-VM/Ctr-to-Phy (NIC-to-VM/Ctr-to-NIC) topology, but due to much higher dependency on intensive memory operations and sensitivity to Linux kernel scheduler settings and behaviour, this estimation may not always yield good enough accuracy.

For detailed FD.io CSIT testbed specification and topology, as well as configuration and setup of SUTs and DUTs testbeds please refer to test_environment.

Similar SUT compute node and DUT VPP settings can be arrived to in a standalone VPP setup by using a vpp-config configuration tool[4] developed within the VPP project using CSIT recommended settings and scripts.

2.1.2 Performance Tests Coverage

Performance tests are split into two main categories:

- Throughput discovery - discovery of packet forwarding rate using binary search in accordance to **RFC 2544**[5].
  - NDR - discovery of Non Drop Rate packet throughput, at zero packet loss; followed by one-way packet latency measurements at 10%, 50% and 100% of discovered NDR throughput.
  - PDR - discovery of Partial Drop Rate, with specified non-zero packet loss currently set to 0.5%; followed by one-way packet latency measurements at 100% of discovered PDR throughput.
- Throughput verification - verification of packet forwarding rate against previously discovered throughput rate. These tests are currently done against 0.9 of reference NDR, with reference rates updated periodically.

CSIT rls1801_2 includes following performance test suites, listed per NIC type:

- 2port10GE X520-DA2 Intel
  - **L2XC** - L2 Cross-Connect switched-forwarding of untagged, dot1q, dot1ad VLAN tagged Ethernet frames.
  - **L2BD** - L2 Bridge-Domain switched-forwarding of untagged Ethernet frames with MAC learning; disabled MAC learning i.e. static MAC tests to be added.
  - **L2BD Scale** - L2 Bridge-Domain switched-forwarding of untagged Ethernet frames with MAC learning; disabled MAC learning i.e. static MAC tests to be added with 20k, 200k and 2M FIB entries.
  - **IPv4** - IPv4 routed-forwarding.
  - **IPv6** - IPv6 routed-forwarding.
  - **IPv4 Scale** - IPv4 routed-forwarding with 20k, 200k and 2M FIB entries.
  - **IPv6 Scale** - IPv6 routed-forwarding with 20k, 200k and 2M FIB entries.
  - **VMs with vhost-user** - virtual topologies with 1 VM and service chains of 2 VMs using vhost-user interfaces, with VPP forwarding modes incl. L2 Cross-Connect, L2 Bridge-Domain, VXLAN with L2BD, IPv4 routed-forwarding.
  - **COP** - IPv4 and IPv6 routed-forwarding with COP address security.
  - **ACL** - L2 Bridge-Domain switched-forwarding and IPv4 and IPv6 routed- forwarding with iACL and oACL IP address, MAC address and L4 port security.
  - **LISP** - LISP overlay tunneling for IPv4-over-IPv4, IPv6-over-IPv4, IPv6-over-IPv6, IPv4-over-IPv6 in IPv4 and IPv6 routed-forwarding modes.
  - **VXLAN** - VXLAN overlay tunnelling integration with L2XC and L2BD.
  - **QoS Policer** - ingress packet rate measuring, marking and limiting (IPv4).

[4] https://wiki.fd.io/view/VPP/Configuration_Tool
[5] https://tools.ietf.org/html/rfc2544.html

  - **NAT** - (Source) Network Address Translation tests with varying number of users and ports per user.
  - **Container memif connections** - VPP memif virtual interface tests to interconnect VPP instances with L2XC and L2BD.
  - **Container K8s Orchestrated Topologies** - Container topologies connected over the memif virtual interface.
  - **SRv6** - Segment Routing IPv6 tests.
- 2port40GE XL710 Intel
  - **L2XC** - L2 Cross-Connect switched-forwarding of untagged Ethernet frames.
  - **L2BD** - L2 Bridge-Domain switched-forwarding of untagged Ethernet frames with MAC learning.
  - **IPv4** - IPv4 routed-forwarding.
  - **IPv6** - IPv6 routed-forwarding.
  - **VMs with vhost-user** - virtual topologies with 1 VM and service chains of 2 VMs using vhost-user interfaces, with VPP forwarding modes incl. L2 Cross-Connect, L2 Bridge-Domain, VXLAN with L2BD, IPv4 routed-forwarding.
  - **IPSecSW** - IPSec encryption with AES-GCM, CBC-SHA1 ciphers, in combination with IPv4 routed-forwarding.
  - **IPSecHW** - IPSec encryption with AES-GCM, CBC-SHA1 ciphers, in combination with IPv4 routed-forwarding. Intel QAT HW acceleration.
  - **IPSec+LISP** - IPSec encryption with CBC-SHA1 ciphers, in combination with LISP-GPE overlay tunneling for IPv4-over-IPv4.
  - **VPP TCP/IP stack** - tests of VPP TCP/IP stack used with VPP built-in HTTP server.
- 2port10GE X710 Intel
  - **L2BD** - L2 Bridge-Domain switched-forwarding of untagged Ethernet frames with MAC learning.
  - **VMs with vhost-user** - virtual topologies with 1 VM using vhost-user interfaces, with VPP forwarding modes incl. L2 Bridge-Domain.
- 2port10GE VIC1227 Cisco
  - **L2BD** - L2 Bridge-Domain switched-forwarding of untagged Ethernet frames with MAC learning.
- 2port40GE VIC1385 Cisco
  - **L2BD - L2 Bridge-Domain switched-forwarding of untagged Ethernet frames** with MAC learning.

Execution of performance tests takes time, especially the throughput discovery tests. Due to limited HW testbed resources available within FD.io labs hosted by LF, the number of tests for NICs other than X520 (a.k.a. Niantic) has been limited to few baseline tests. CSIT team expect the HW testbed resources to grow over time, so that complete set of performance tests can be regularly and(or) continuously executed against all models of hardware present in FD.io labs.

2.1.3 Performance Tests Naming

CSIT rls1801_2 follows a common structured naming convention for all performance and system functional tests, introduced in CSIT rls1710.

The naming should be intuitive for majority of the tests. Complete description of CSIT test naming convention is provided on CSIT test naming wiki[6].

2.1.4 Methodology: Multi-Core and Multi-Threading

Intel Hyper-Threading - CSIT rls1801_2 performance tests are executed with SUT servers' Intel XEON processors configured in Intel Hyper-Threading Disabled mode (BIOS setting). This is the simplest configuration used to establish baseline single-thread single-core application packet processing and forwarding performance. Subsequent releases of CSIT will add performance tests with Intel Hyper-Threading Enabled (requires BIOS settings change and hard reboot of server).

Multi-core Tests - CSIT rls1801_2 multi-core tests are executed in the following VPP thread and core configurations:

1. 1t1c - 1 VPP worker thread on 1 CPU physical core.
2. 2t2c - 2 VPP worker threads on 2 CPU physical cores.
3. 4t4c - 4 VPP worker threads on 4 CPU physical cores.

VPP worker threads are the data plane threads. VPP control thread is running on a separate non-isolated core together with other Linux processes. Note that in quite a few test cases running VPP workers on 2 or 4 physical cores hits the I/O bandwidth or packets-per-second limit of tested NIC.

Section *Throughput Speedup Multi-Core* (page 62) includes a set of graphs illustrating packet throughout speedup when running VPP on multiple cores.

2.1.5 Methodology: Packet Throughput

Following values are measured and reported for packet throughput tests:

- NDR binary search per **RFC 2544**[7]:
  - Packet rate: "RATE: <aggregate packet rate in packets-per-second> pps (2x <per direction packets-per-second>)";
  - Aggregate bandwidth: "BANDWIDTH: <aggregate bandwidth in Gigabits per second> Gbps (untagged)";
- PDR binary search per **RFC 2544**[8]:
  - Packet rate: "RATE: <aggregate packet rate in packets-per-second> pps (2x <per direction packets-per-second>)";
  - Aggregate bandwidth: "BANDWIDTH: <aggregate bandwidth in Gigabits per second> Gbps (untagged)";
  - Packet loss tolerance: "LOSS_ACCEPTANCE <accepted percentage of packets lost at PDR rate>";
- NDR and PDR are measured for the following L2 frame sizes:
  - IPv4: 64B, IMIX_v4_1 (28x64B,16x570B,4x1518B), 1518B, 9000B;
  - IPv6: 78B, 1518B, 9000B;
- NDR and PDR binary search resolution is determined by the final value of the rate change, referred to as the final step:
  - The final step is set to 50kpps for all NIC to NIC tests and all L2 frame sizes except 9000B (changed from 100kpps used in previous releases).

[6] https://wiki.fd.io/view/CSIT/csit-test-naming
[7] https://tools.ietf.org/html/rfc2544.html
[8] https://tools.ietf.org/html/rfc2544.html

  - The final step is set to 10kpps for all remaining tests, including 9000B and all vhost VM and memif Container tests.

All rates are reported from external Traffic Generator perspective.

2.1.6 Methodology: Packet Latency

TRex Traffic Generator (TG) is used for measuring latency of VPP DUTs. Reported latency values are measured using following methodology:

- Latency tests are performed at 10%, 50% of discovered NDR rate (non drop rate) for each NDR throughput test and packet size (except IMIX).
- TG sends dedicated latency streams, one per direction, each at the rate of 10kpps at the prescribed packet size; these are sent in addition to the main load streams.
- TG reports min/avg/max latency values per stream direction, hence two sets of latency values are reported per test case; future release of TRex is expected to report latency percentiles.
- Reported latency values are aggregate across two SUTs due to three node topology used for all performance tests; for per SUT latency, reported value should be divided by two.
- 1usec is the measurement accuracy advertised by TRex TG for the setup used in FD.io labs used by CSIT project.
- TRex setup introduces an always-on error of about 2*2usec per latency flow - additonal Tx/Rx interface latency induced by TRex SW writing and reading packet timestamps on CPU cores without HW acceleration on NICs closer to the interface line.

2.1.7 Methodology: KVM VM vhost

CSIT rls1801_2 introduced test environment configuration changes to KVM Qemu vhost-user tests in order to more representatively measure VPP-18.01.2 release performance in configurations with vhost-user interfaces and different Qemu settings.

FD.io CSIT performance lab is testing VPP vhost with KVM VMs using following environment settings:

- Tests with varying Qemu virtio queue (a.k.a. vring) sizes: [vr256] default 256 descriptors, [vr1024] 1024 descriptors to optimize for packet throughput;
- Tests with varying Linux CFS (Completely Fair Scheduler) settings: [cfs] default settings, [cfsrr1] CFS RoundRobin(1) policy applied to all data plane threads handling test packet path including all VPP worker threads and all Qemu testpmd poll-mode threads;
- Resulting test cases are all combinations with [vr256,vr1024] and [cfs,cfsrr1] settings;
- Adjusted Linux kernel CFS scheduler policy for data plane threads used in CSIT is documented in CSIT Performance Environment Tuning wiki[9]. The purpose is to verify performance impact (NDR, PDR throughput) and same test measurements repeatability, by making VPP and VM data plane threads less susceptible to other Linux OS system tasks hijacking CPU cores running those data plane threads.

2.1.8 Methodology: LXC and Docker Containers memif

CSIT rls1801_2 introduced additional tests taking advantage of VPP memif virtual interface (shared memory interface) tests to interconnect VPP instances. VPP vswitch instance runs in bare-metal user-mode handling Intel x520 NIC 10GbE interfaces and connecting over memif (Master side) virtual interfaces to more instances of VPP running in LXC (Linux Container) or in Docker Containers, both with memif virtual interfaces (Slave side). LXCs and Docker Containers run in a priviliged mode with VPP data plane worker threads pinned to dedicated physical CPU cores per usual CSIT practice. All VPP instances run the

[9] https://wiki.fd.io/view/CSIT/csit-perf-env-tuning-ubuntu1604

same version of software. This test topology is equivalent to existing tests with vhost-user and VMs as described earlier in *Tested Physical Topologies* (page 4).

More information about CSIT LXC and Docker Container setup and control is available in *Container Orchestration in CSIT* (page 157).

2.1.9 Methodology: Container Topologies Orchestrated by K8s

CSIT rls1801_2 introduced new tests of Container topologies connected over the memif virtual interface (shared memory interface). In order to provide simple topology coding flexibility and extensibility container orchestration is done with Kubernetes[10] using Docker[11] images for all container applications including VPP. Ligato[12] is used to address the container networking orchestration that is integrated with K8s, including memif support.

For these tests VPP vswitch instance runs in a Docker Container handling Intel x520 NIC 10GbE interfaces and connecting over memif (Master side) virtual interfaces to more instances of VPP running in Docker Containers with memif virtual interfaces (Slave side). All Docker Containers run in a priviliged mode with VPP data plane worker threads pinned to dedicated physical CPU cores per usual CSIT practice. All VPP instances run the same version of software. This test topology is equivalent to existing tests with vhost-user and VMs as described earlier in *Tested Physical Topologies* (page 4).

More information about CSIT Container Topologies Orchestrated by K8s is available in *Container Orchestration in CSIT* (page 157).

2.1.10 Methodology: IPSec with Intel QAT HW cards

VPP IPSec performance tests are using DPDK cryptodev device driver in combination with HW cryptodev devices - Intel QAT 8950 50G - present in LF FD.io physical testbeds. DPDK cryptodev can be used for all IPSec data plane functions supported by VPP.

Currently CSIT rls1801_2 implements following IPSec test cases:

- AES-GCM, CBC-SHA1 ciphers, in combination with IPv4 routed-forwarding with Intel xl710 NIC.
- CBC-SHA1 ciphers, in combination with LISP-GPE overlay tunneling for IPv4-over-IPv4 with Intel xl710 NIC.

2.1.11 Methodology: TRex Traffic Generator Usage

TRex traffic generator[13] is used for all CSIT performance tests. TRex stateless mode is used to measure NDR and PDR throughputs using binary search (NDR and PDR discovery tests) and for quick checks of DUT performance against the reference NDRs (NDR check tests) for specific configuration.

TRex is installed and run on the TG compute node. The typical procedure is:

- If the TRex is not already installed on TG, it is installed in the suite setup phase - see TRex intallation[14].
- TRex configuration is set in its configuration file

```
/etc/trex_cfg.yaml
```

- TRex is started in the background mode

[10] https://github.com/kubernetes
[11] https://github.com/docker
[12] https://github.com/ligato
[13] https://wiki.fd.io/view/TRex
[14] https://git.fd.io/csit/tree/resources/tools/trex/trex_installer.sh?h=rls1801_2

```
$ sh -c 'cd <t-rex-install-dir>/scripts/ && sudo nohup ./t-rex-64 -i -c 7 --iom 0 > /tmp/trex.
↪log 2>&1 &' > /dev/null
```

- There are traffic streams dynamically prepared for each test, based on traffic profiles. The traffic is sent and the statistics obtained using **`trex_stl_lib.api.STLClient`**.

Measuring packet loss

- Create an instance of STLClient
- Connect to the client
- Add all streams
- Clear statistics
- Send the traffic for defined time
- Get the statistics

If there is a warm-up phase required, the traffic is sent also before test and the statistics are ignored.

Measuring latency

If measurement of latency is requested, two more packet streams are created (one for each direction) with TRex flow_stats parameter set to STLFlowLatencyStats. In that case, returned statistics will also include min/avg/max latency values.

2.1.12 Methodology: TCP/IP tests with WRK tool

WRK HTTP benchmarking tool[15] is used for experimental TCP/IP and HTTP tests of VPP TCP/IP stack and built-in static HTTP server. WRK has been chosen as it is capable of generating significant TCP/IP and HTTP loads by scaling number of threads across multi-core processors.

This in turn enables quite high scale benchmarking of the main TCP/IP and HTTP service including HTTP TCP/IP Connections-Per-Second (CPS), HTTP Requests-Per-Second and HTTP Bandwidth Throughput.

The initial tests are designed as follows:

- HTTP and TCP/IP Connections-Per-Second (CPS)
  - WRK configured to use 8 threads across 8 cores, 1 thread per core.
  - Maximum of 50 concurrent connections across all WRK threads.
  - Timeout for server responses set to 5 seconds.
  - Test duration is 30 seconds.
  - Expected HTTP test sequence:
    - Single HTTP GET Request sent per open connection.
    - Connection close after valid HTTP reply.
    - Resulting flow sequence - 8 packets: >S,<S-A,>A,>Req,<Rep,>F,<F,> A.
- HTTP Requests-Per-Second
  - WRK configured to use 8 threads across 8 cores, 1 thread per core.
  - Maximum of 50 concurrent connections across all WRK threads.
  - Timeout for server responses set to 5 seconds.
  - Test duration is 30 seconds.
  - Expected HTTP test sequence:

[15] https://github.com/wg/wrk

* Multiple HTTP GET Requests sent in sequence per open connection.
* Connection close after set test duration time.
* Resulting flow sequence: >S,<S-A,>A,>Req[1],<Rep[1],..,>Req[n],<Rep[n],>F,<F,>A.

2.2 CSIT Release Notes

2.2.1 Changes in CSIT rls1801_2

1. Added VPP performance tests
   - **Container Service Chain Topologies Orchestrated by K8s with VPP Memif**
     - Added tests with VPP vswitch in container connecting a number of VPP- in-container service chain topologies with L2 Cross-Connect and L2 Bridge-Domain configurations, orchestrated by Kubernetes. Added following forwarding topologies: i) "Parallel" with packets flowing from NIC via VPP to container and back to VPP and NIC; ii) "Chained" (a.k.a. "Snake") with packets flowing via VPP to container, back to VPP, to next container, back to VPP and so on until the last container in a chain, then back to VPP and NIC; iii) "Horizontal" with packets flowing via VPP to container, then via "horizontal" memif to next container, and so on until the last container, then back to VPP and NIC;
   - **VPP TCP/IP stack**
     - Added tests for VPP TCP/IP stack using VPP built-in HTTP server. WRK traffic generator is used as a client-side;
   - **SRv6**
     - Initial SRv6 (Segment Routing IPv6) tests verifying performance of IPv6 and SRH (Segment Routing Header) encapsulation, decapsulation, lookups and rewrites based on configured End and End.DX6 SRv6 egress functions;
   - **IPSecSW**
     - SW computed IPSec encryption with AES-GCM, CBC-SHA1 ciphers, in combination with IPv4 routed-forwarding;
2. Presentation and Analytics Layer
   - Added throughput speedup analysis for multi-core and multi-thread VPP tests into Presentation and Analytics Layer (PAL) for automated CSIT test results analysis;
3. Other changes
   - **Framework optimizations**
     - Ability to run CSIT framework on ARM architecture;
     - Overall stability improvements;
   - **NDR and PDR throughput binary search change**
     - Increased binary search resolution by reducing final step from 100kpps to 50kpps;
   - **VPP plugin loaded as needed by tests**
     - From this release only plugins required by tests are loaded at VPP initialization time. Previously all plugins were loaded for all tests;

2.2.2 Performance Changes

Relative performance changes in measured packet throughput in CSIT rls1801_2 are calculated against the results from CSIT rls1801 report. Listed mean and standard deviation values are computed based on

a series of the same tests executed against respective VPP releases to verify test results repeatibility, with percentage change calculated for mean values. Note that the standard deviation is quite high for a small number of packet throughput tests, what indicates poor test results repeatability and makes the relative change of mean throughput value not fully representative for these tests. The root causes behind poor results repeatibility vary between the test cases.

NDR Throughput Changes

NDR small packet throughput changes between releases are available in a CSV and pretty ASCII formats:

- csv format for 1t1c,
- csv format for 2t2c,
- pretty ASCII format for 1t1c,
- pretty ASCII format for 2t2c.

PDR Throughput Changes

NDR small packet throughput changes between releases are available in a CSV and pretty ASCII formats:

- csv format for 1t1c,
- csv format for 2t2c,
- pretty ASCII format for 1t1c,
- pretty ASCII format for 2t2c.

Measured improvements are in line with VPP code optimizations listed in VPP-18.01 release notes[16].

2.2.3 Known Issues

Here is the list of known issues in CSIT rls1801_2 for VPP performance tests:

[16] https://docs.fd.io/vpp/18.01/release_notes_1801.html

#	Issue	Jira ID	Description
1	Vic1385 and Vic1227 low performance.	VPP-664	Low NDR performance.
2	Sporadic (1 in 200) NDR discovery test failures on x520.	CSIT-570	DPDK reporting rx-errors, indicating L1 issue. Suspected issue with HW combination of X710-X520 in LF testbeds. Not observed outside of LF testbeds.
3	Lower than expected NDR throughput with xl710 and x710 NICs, compared to x520 NICs.	CSIT-571	Suspected NIC firmware or DPDK driver issue affecting NDR and PDR throughput. Applies to XL710 and X710 NICs.
4	QAT IPSec scale with 1000 tunnels (interfaces) in 2t2c config, all tests are failing.	VPP-1121	VPP crashes during configuration of 1000 IPsec tunnels. 1t1c tests are not affected
5	rls1801 plugin related performance regression	CSIT-925	With all plugins loaded NDR, PDR and MaxRates vary intermittently from 3% to 5% across multiple test executions. Requires plugin code bisecting.
6	rls1801 generic small performance regression ip4base, l2xcbase, l2bdbase	CSIT-926	Generic performance regression of discovered NDR, PDR and MaxRates of -3%..-1% vs. rls1710, affects ip4base, l2xcbase, l2bdbase test suites. Not detected by CSIT performance trending scheme as it was masked out by another issue CSIT-925.
7	rls1801 substantial NDR performance regression for vhost-user vring size of 1024	CSIT-927	Much lower NDR for vhostvr1024 tests, with mean values regression of -17%..-42% vs. rls1710, but also very high standard deviation of up to 1.46 Mpps => poor repeatibility. Making mean values not fully representative.
8	rls1801 substantial NDR/PDR regression for IPSec tunnel scale with HW QAT crypto-dev	CSIT-928	NDR regression of -7%..-15%, PDR regression of -3%..-15% compared to rls1710.
9	NAT plugin requires ACL plugin enabled in VPP release 18.01.2	N/A	Because of nodes order on interface feature all tests related to NAT plugin needs to have enabled ACL plugin too.

2.3 Packet Throughput Graphs

Plotted results are generated by multiple executions of the same CSIT performance tests across three physical testbeds within LF FD.io labs. To provide a descriptive summary view, Box-and-Whisker plots are used to display variation in measured throughput values, without making any assumptions of the underlying statistical distribution.

For each plotted test case, Box-and-Whisker plots show the quartiles (Min, 1st quartile / 25th percentile, 2nd quartile / 50th percentile / mean, 3rd quartile / 75th percentile, Max) across collected data set (data set size stated in the note below). Outliers are plotted as individual points. Min and max values are plotted as bottom and top Whiskers respectively. 2nd and 3rd quartiles are plotted as bottom and top edge of the box. If multiple samples match only two values, and all samples fall between them, then no whiskers are plotted. If all samples have the same value, only a horizontal line is plotted.

Title of each graph is a regex (regular expression) matching all throughput test cases plotted on this graph, *X-axis labels* are indices of individual test suites executed by FD.io test executor vpp performance jobs[17] jobs that created result output files used as data sources for the graph, *Y-axis labels* are measured Packets Per Second [pps] values, and the *Graph legend* lists the plotted test suites and their indices.

[17] https://jenkins.fd.io/view/csit/job/csit-vpp-perf-1801-all

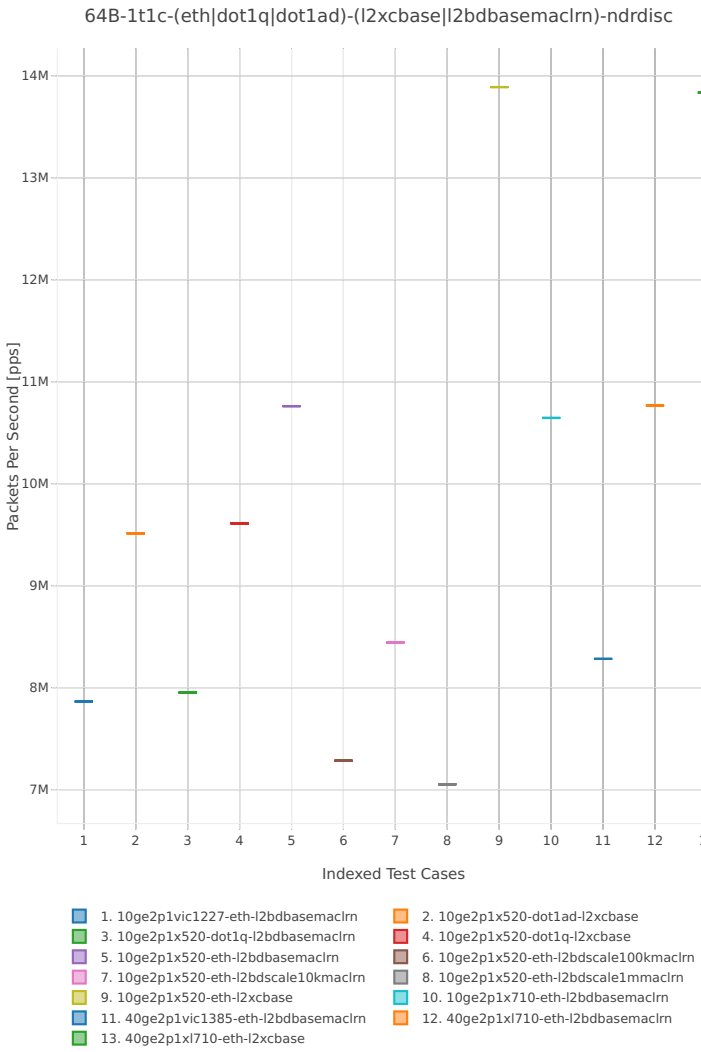
Note: Test results have been generated by FD.io test executor vpp performance jobs[18] with Robot Framework result files csit-vpp-perf-1801-*.zip archived here. Plotted data set size per test case is equal to the number of job executions presented in this report version: **10**.

2.3.1 L2 Ethernet Switching

Following sections include summary graphs of VPP Phy-to-Phy performance with L2 Ethernet switching, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss). Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

NDR Throughput

VPP NDR 64B packet throughput in 1t1c setup (1thread, 1core) is presented in the graph below.



CSIT source code for the test cases used for above plots can be found in CSIT git repository:

[18] https://jenkins.fd.io/view/csit/job/csit-vpp-perf-1801-all

```
$ grep -E "64B-1t1c-(eth|dot1q|dot1ad)-(l2xcbase|l2bdbasemaclrn|l2bdscale.*|l2dbscale.*)-(eth.
↪*)*ndrdisc" tests/vpp/perf/l2/*
```

Figure 1a. VPP 1thread 1core - NDR Throughput for Phy-to-Phy L2 Ethernet Switching (base).

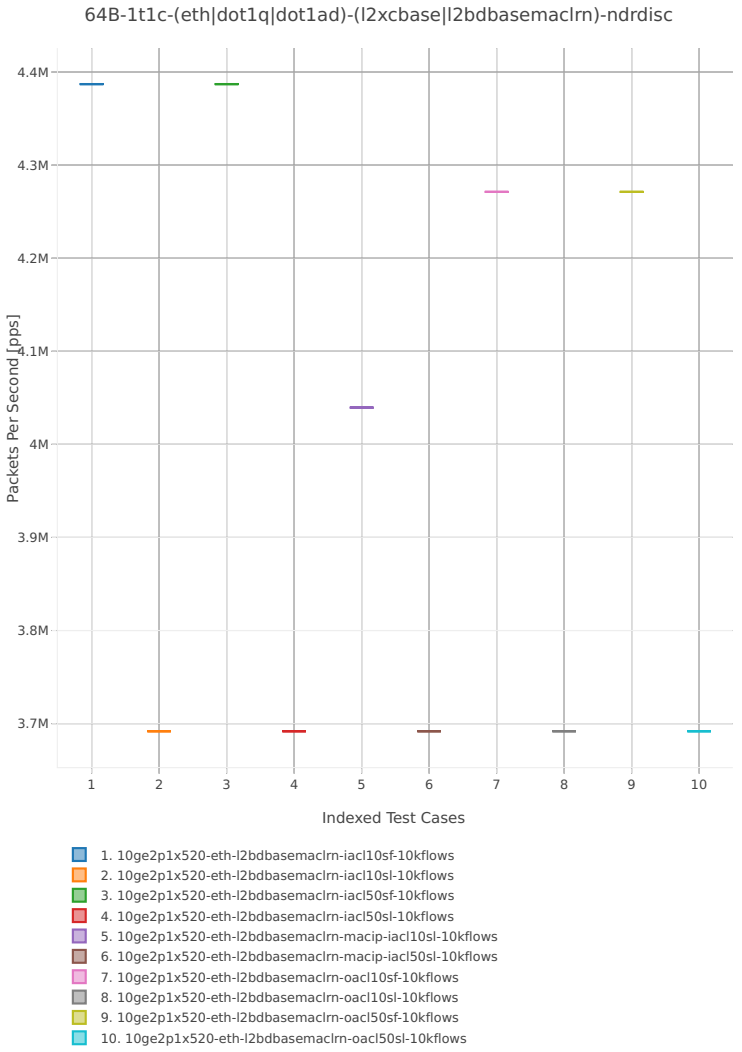
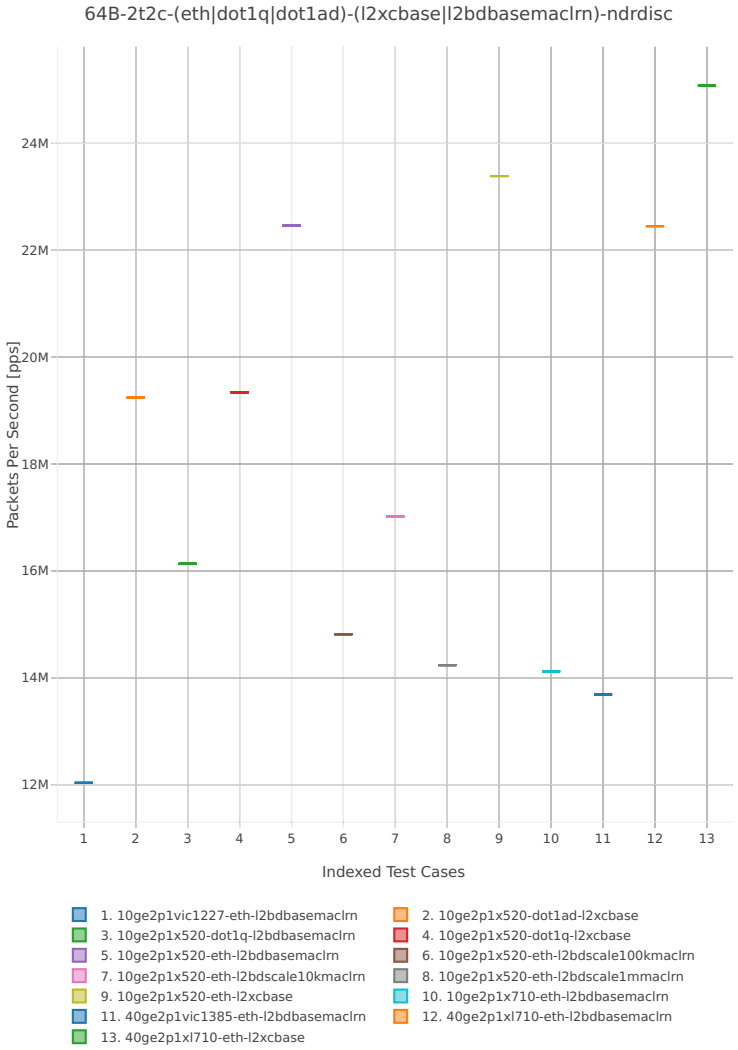


Figure 1b. VPP 1thread 1core - NDR Throughput for Phy-to-Phy L2 Ethernet Switching (feature).

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/l2
$ grep -E "64B-1t1c-(eth|dot1q|dot1ad)-(l2xcbase|l2bdbasemaclrn).*-(iacl50(-state(ful|less)|sl)-
↪(flows10k.*|10kflows.*)|oacl50-state(ful|less)-flows10k.*)-ndrdisc" *
```

VPP NDR 64B packet throughput in 2t2c setup (2thread, 2core) is presented in the graph below.



CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ grep -E "64B-2t2c-(eth|dot1q|dot1ad)-(l2xcbase|l2bdbasemaclrn|l2bdscale.*|l2dbscale.*)-(eth.
↪*)*ndrdisc" tests/vpp/perf/l2/*
```

Figure 2a. VPP 2threads 2cores - NDR Throughput for Phy-to-Phy L2 Ethernet Switching (base).

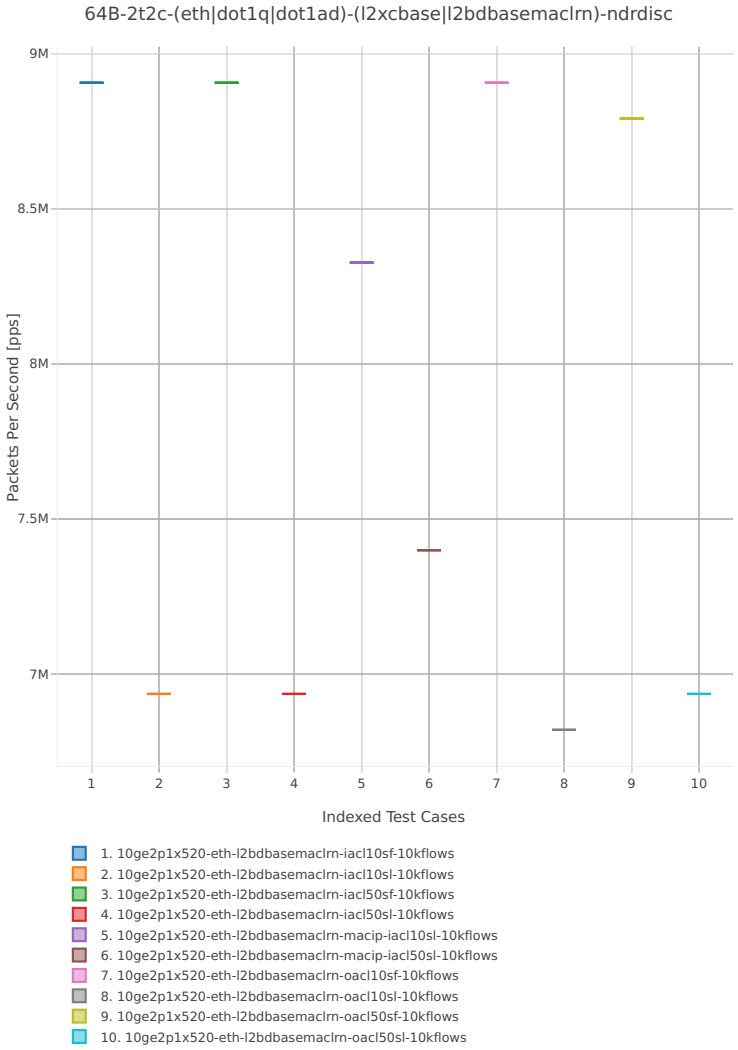


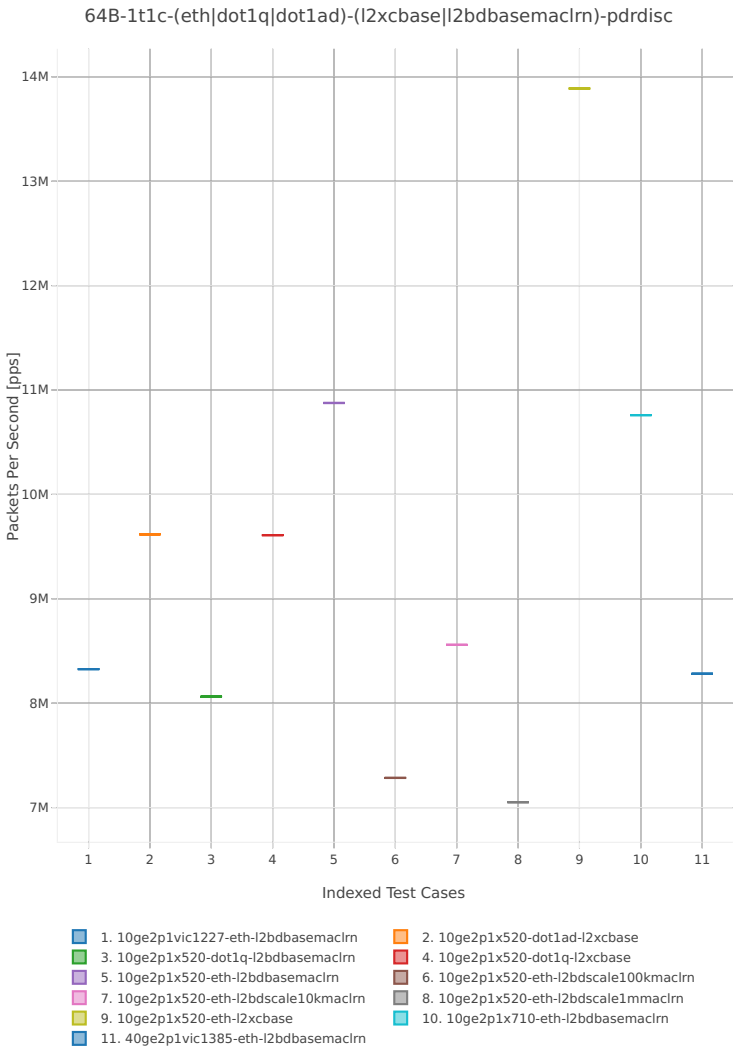
Figure 2b. VPP 2threads 2cores - NDR Throughput for Phy-to-Phy L2 Ethernet Switching (feature).

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/l2
$ grep -E "64B-2t2c-(eth|dot1q|dot1ad)-(l2xcbase|l2bdbasemaclrn).*-(iacl50(-state(ful|less)|sl)-
↪(flows10k.*|10kflows.*)|oacl50-state(ful|less)-flows10k.*)-ndrdisc" *
```

PDR Throughput

VPP PDR 64B packet throughput in 1t1c setup (1thread, 1core) is presented in the graph below. PDR measured for 0.5% packet loss ratio.



CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ grep -E "64B-1t1c-(eth|dot1q|dot1ad)-(l2xcbase|l2bdbasemaclrn|l2bdscale.*|l2dbscale.*)-(eth.
↪*)*ndrdisc" tests/vpp/perf/l2/*
```

Figure 3a. VPP 1thread 1core - PDR Throughput for Phy-to-Phy L2 Ethernet Switching (base).

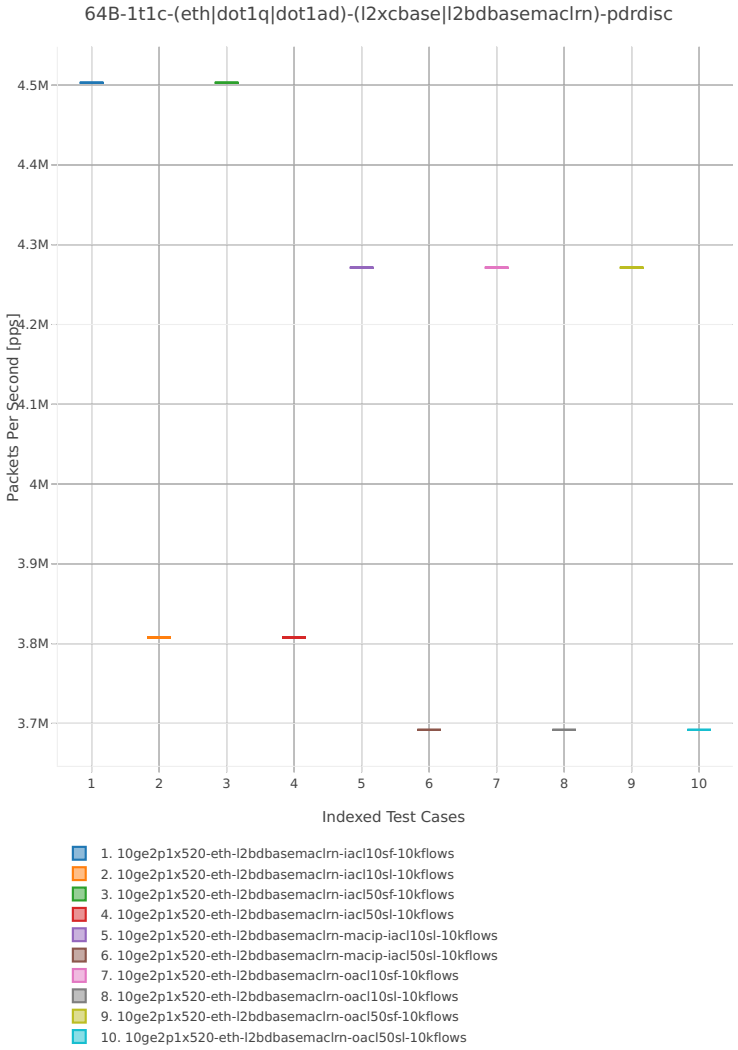
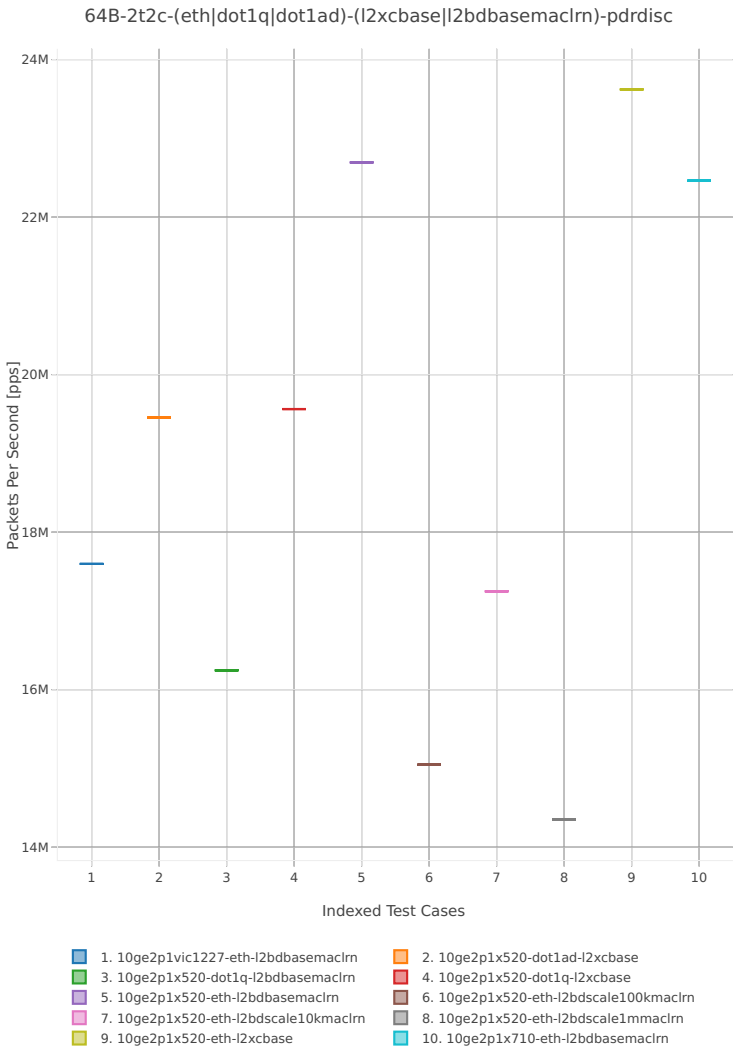


Figure 3b. VPP 1thread 1core - PDR Throughput for Phy-to-Phy L2 Ethernet Switching (feature).

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/l2
$ grep -E "64B-1t1c-(eth|dot1q|dot1ad)-(l2xcbase|l2bdbasemaclrn).*-(iacl50(-state(ful|less)|sl)-
↪(flows10k.*|10kflows.*)|oacl50-state(ful|less)-flows10k.*)-pdrdisc" *
```

VPP PDR 64B packet throughput in 2t2c setup (2thread, 2core) is presented in the graph below. PDR measured for 0.5% packet loss ratio.



CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ grep -E "64B-2t2c-(eth|dot1q|dot1ad)-(l2xcbase|l2bdbasemaclrn|l2bdscale.*|l2dbscale.*)-(eth.
↪*)*ndrdisc" tests/vpp/perf/l2/*
```

Figure 4a. VPP 2thread 2core - PDR Throughput for Phy-to-Phy L2 Ethernet Switching (base).

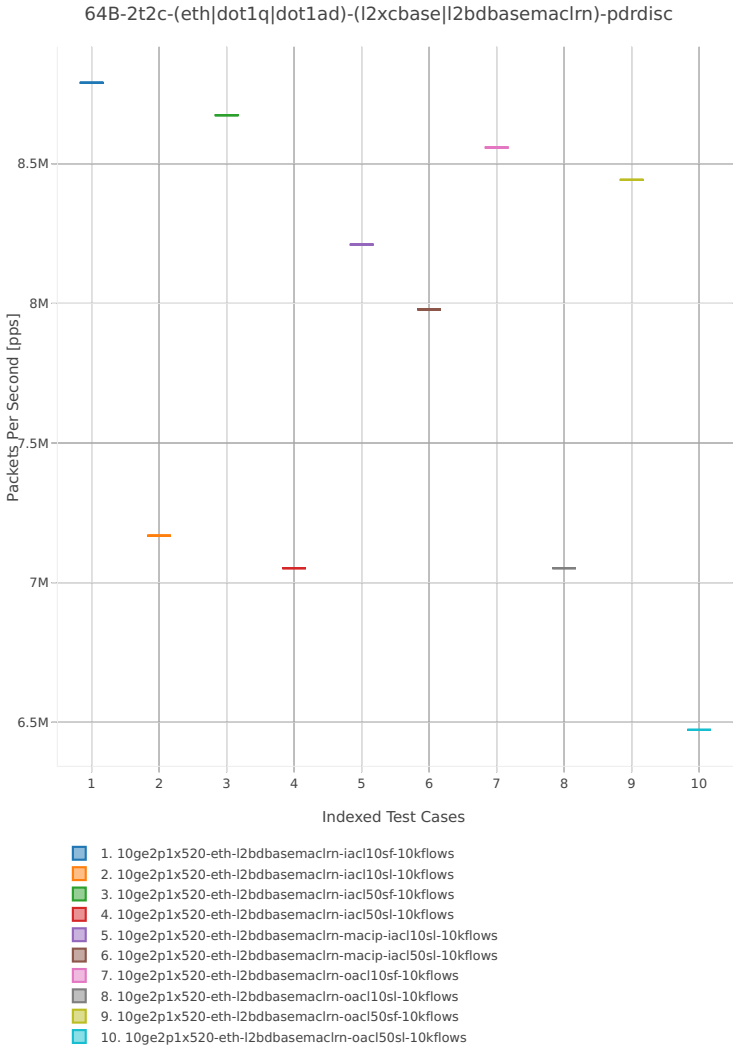


Figure 4b. VPP 2thread 2core - PDR Throughput for Phy-to-Phy L2 Ethernet Switching (feature).

```
$ cd tests/vpp/perf/l2
$ grep -E "64B-2t2c-(eth|dot1q|dot1ad)-(l2xcbase|l2bdbasemaclrn).*-(iacl50(-state(ful|less)|sl)-
↪(flows10k.*|10kflows.*)|oacl50-state(ful|less)-flows10k.*)-pdrdisc" *
```

2.3.2 IPv4 Routed-Forwarding

Following sections include summary graphs of VPP Phy-to-Phy performance with IPv4 Routed-Forwarding, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss). Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

NDR Throughput

VPP NDR 64B packet throughput in 1t1c setup (1thread, 1core) is presented in the graph below.

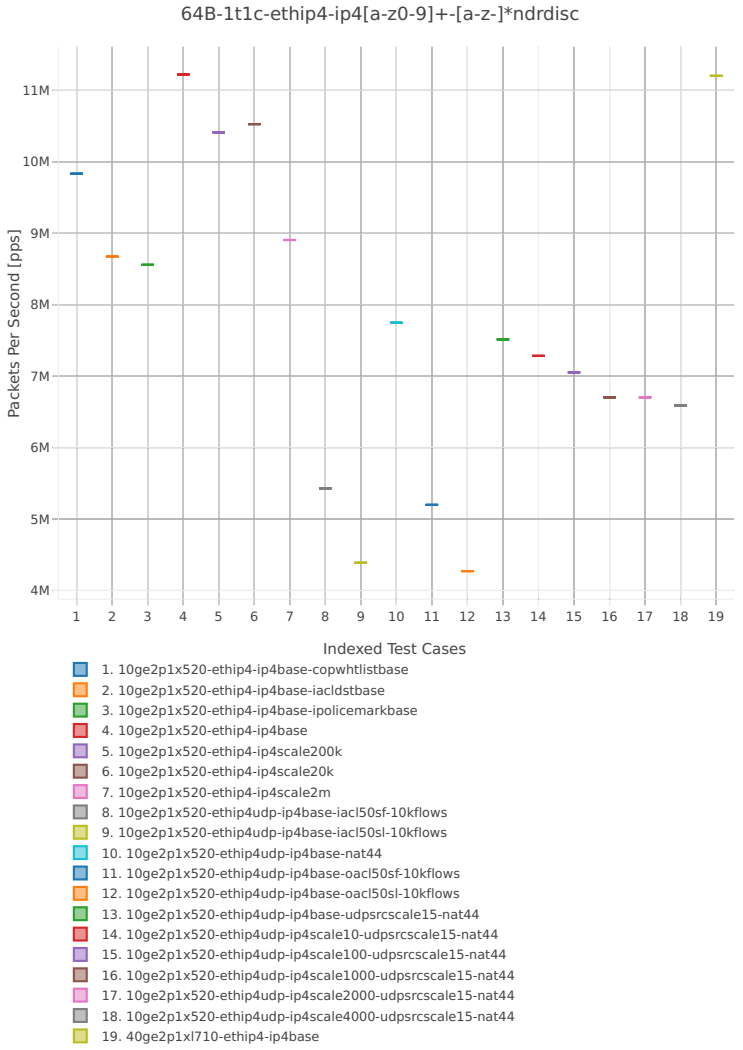


Figure 1. VPP 1thread 1core - NDR Throughput for Phy-to-Phy IPv4 Routed-Forwarding.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/ip4
$ grep -P '64B-1t1c-ethip4(udp)*-ip4(base|scale[a-z0-9]*)(-iacl50-state(ful|less)-flows10k.*|-
↪oacl50-state(ful|less)-flows10k.*|-snat.*|-udp.*|-cop.*|-iacldst.*|-ipolice.*)*-ndrdisc' *
```

VPP NDR 64B packet throughput in 2t2c setup (2thread, 2core) is presented in the graph below.

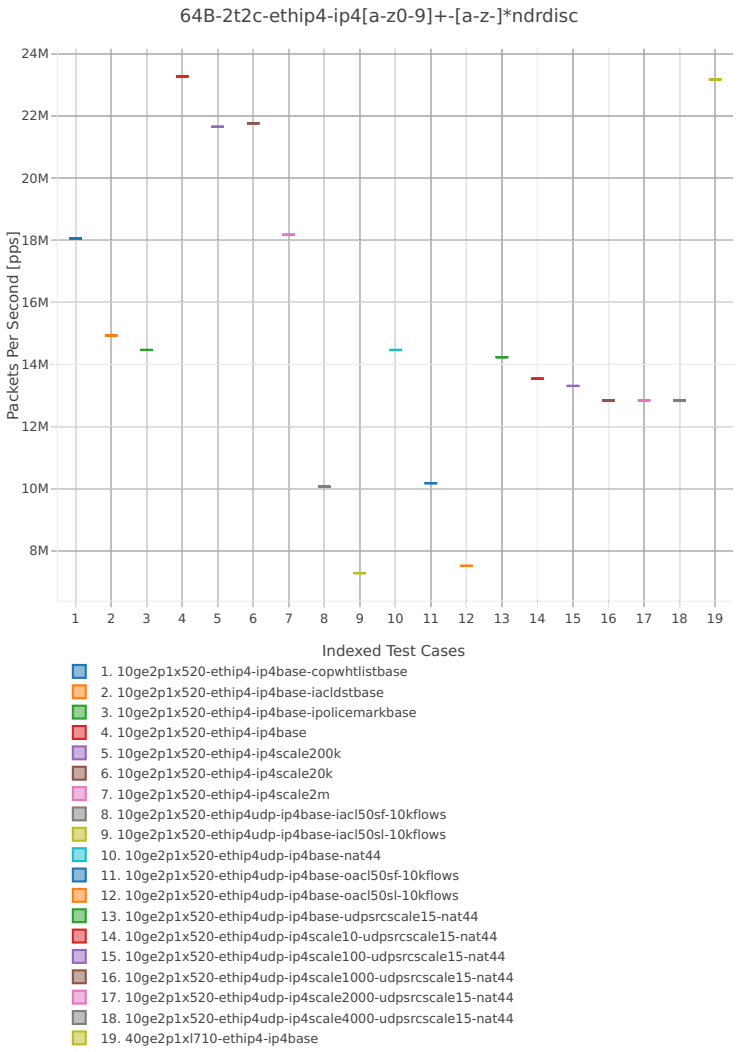


Figure 2. VPP 2threads 2cores - NDR Throughput for Phy-to-Phy IPv4 Routed-Forwarding.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/ip4
$ grep -P '64B-2t2c-ethip4(udp)*-ip4(base|scale[a-z0-9]*)(-iacl50-state(ful|less)-flows10k.*|-
↪oacl50-state(ful|less)-flows10k.*|-snat.*|-udp.*|-cop.*|-iacldst.*|-ipolice.*)*-ndrdisc' *
```

PDR Throughput

VPP PDR 64B packet throughput in 1t1c setup (1thread, 1core) is presented in the graph below. PDR measured for 0.5% packet loss ratio.

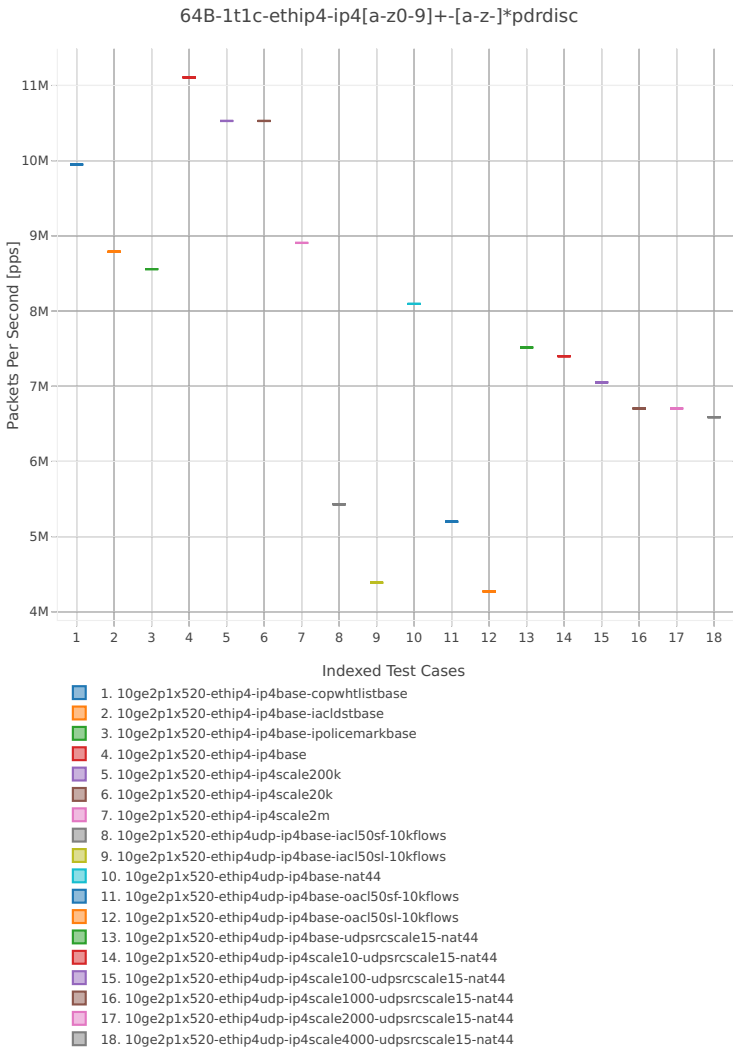


Figure 3. VPP 1thread 1core - PDR Throughput for Phy-to-Phy IPv4 Routed-Forwarding.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/ip4
$ grep -P '64B-1t1c-ethip4(udp)*-ip4(base|scale[a-z0-9]*)(-iacl50-state(ful|less)-flows10k.*|-
↪oacl50-state(ful|less)-flows10k.*|-snat.*|-udp.*|-cop.*|-iacldst.*|-ipolice.*)*-pdrdisc' *
```

VPP PDR 64B packet throughput in 2t2c setup (2thread, 2core) is presented in the graph below. PDR measured for 0.5% packet loss ratio.

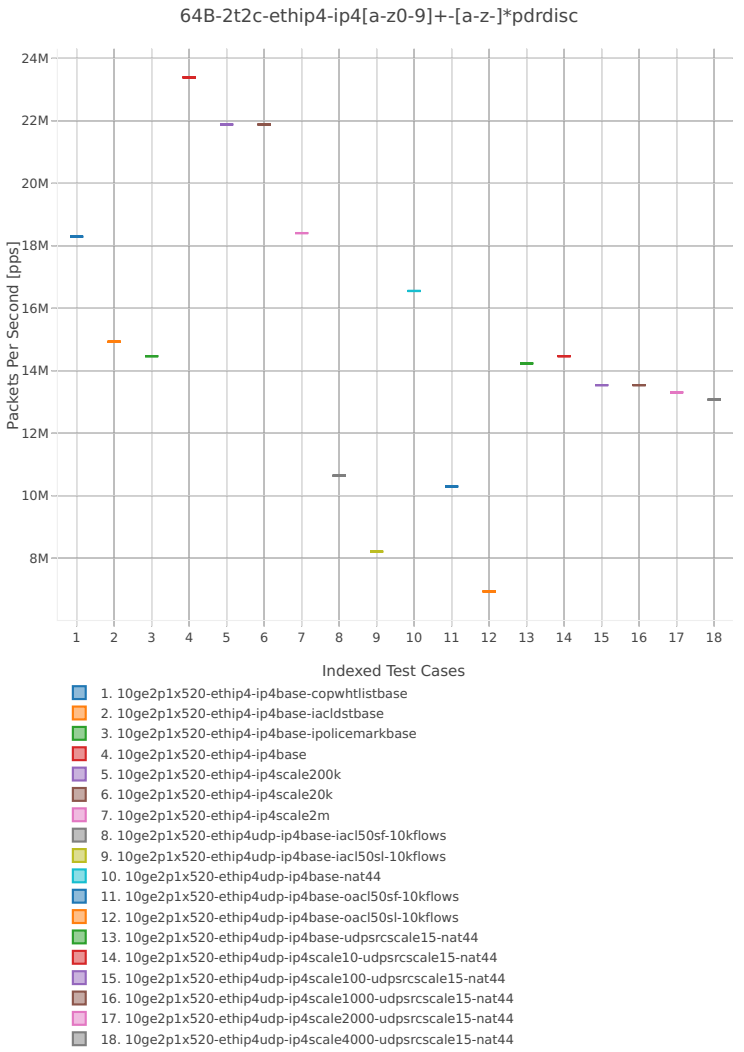


Figure 4. VPP 2thread 2core - PDR Throughput for Phy-to-Phy IPv4 Routed-Forwarding.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/ip4
$ grep -P '64B-2t2c-ethip4(udp)*-ip4(base|scale[a-z0-9]*)(-iacl50-state(ful|less)-flows10k.*|-
↪oacl50-state(ful|less)-flows10k.*|-snat.*|-udp.*|-cop.*|-iacldst.*|-ipolice.*)*-pdrdisc' *
```

2.3.3 IPv6 Routed-Forwarding

Following sections include summary graphs of VPP Phy-to-Phy performance with IPv6 Routed-Forwarding, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss). Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

NDR Throughput

VPP NDR 78B packet throughput in 1t1c setup (1thread, 1core) is presented in the graph below.

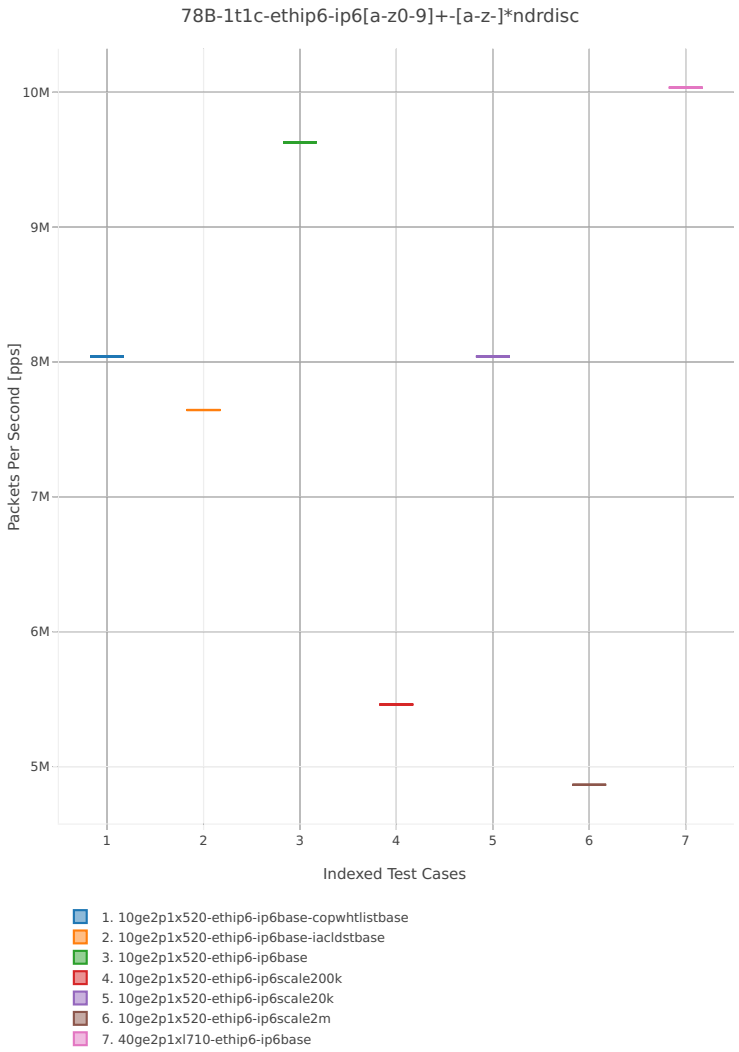


Figure 1. VPP 1thread 1core - NDR Throughput for Phy-to-Phy IPv6 Routed-Forwarding.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/ip6
$ grep -E "78B-1t1c-ethip6-ip6[a-z0-9]+-[a-z-]*ndrdisc" *
```

VPP NDR 78B packet throughput in 2t2c setup (2thread, 2core) is presented in the graph below.

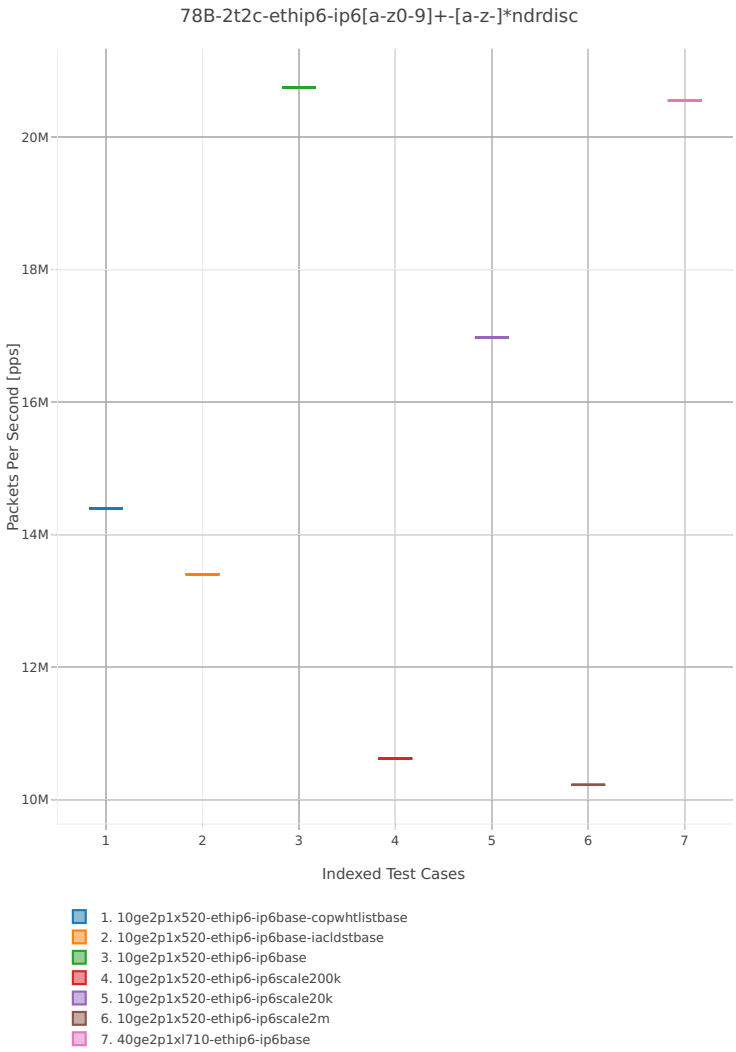


Figure 2. VPP 2threads 2cores - NDR Throughput for Phy-to-Phy IPv6 Routed-Forwarding.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/ip6
$ grep -E "78B-2t2c-ethip6-ip6[a-z0-9]+-[a-z-]*ndrdisc" *
```

PDR Throughput

VPP PDR 78B packet throughput in 1t1c setup (1thread, 1core) is presented in the graph below. PDR measured for 0.5% packet loss ratio.

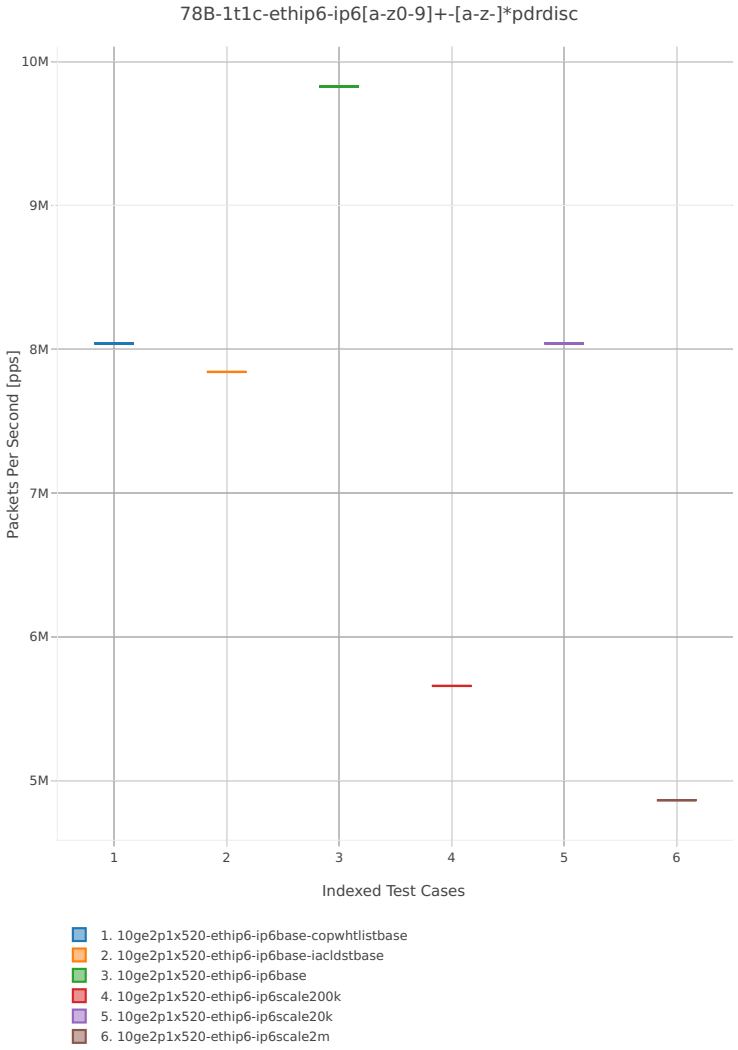


Figure 3. VPP 1thread 1core - PDR Throughput for Phy-to-Phy IPv6 Routed-Forwarding.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/ip6
$ grep -E "78B-1t1c-ethip6-ip6[a-z0-9]+-[a-z-]*pdrdisc" *
```

VPP PDR 78B packet throughput in 2t2c setup (2thread, 2core) is presented in the graph below. PDR measured for 0.5% packet loss ratio.

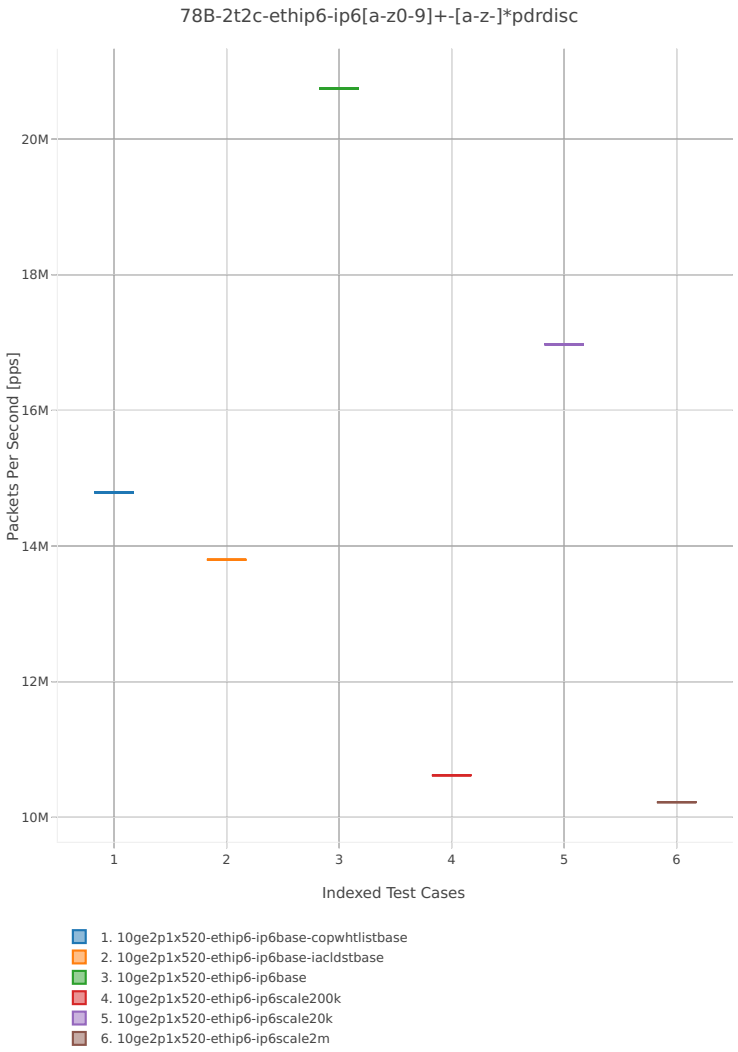


Figure 4. VPP 2thread 2core - PDR Throughput for Phy-to-Phy IPv6 Routed-Forwarding.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/ip6
$ grep -E "78B-2t2c-ethip6-ip6[a-z0-9]+-[a-z-]*pdrdisc" *
```

2.3.4 SRv6

Following sections include summary graphs of VPP Phy-to-Phy performance with SRv6, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss). Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

NDR Throughput

VPP NDR 78B packet throughput in 1t1c setup (1thread, 1core) is presented in the graph below.

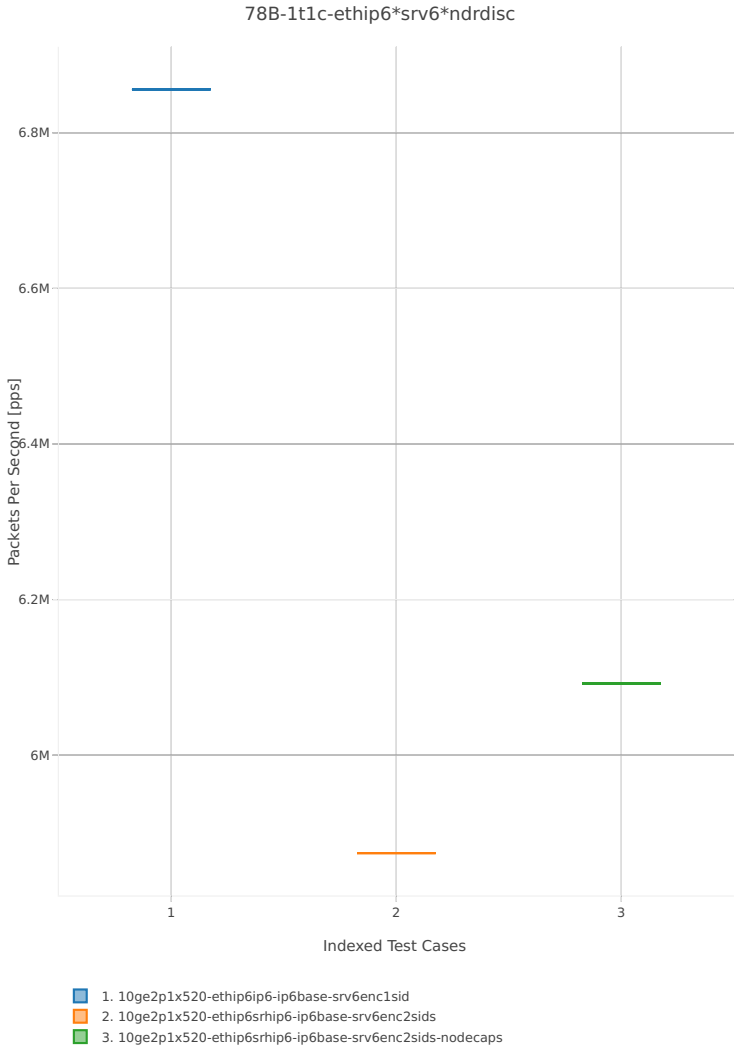


Figure 1. VPP 1thread 1core - NDR Throughput for Phy-to-Phy SRv6.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[19].

VPP NDR 78B packet throughput in 2t2c setup (2thread, 2core) is presented in the graph below.

[19] https://git.fd.io/csit/tree/tests/vpp/perf/srv6?h=rls1804

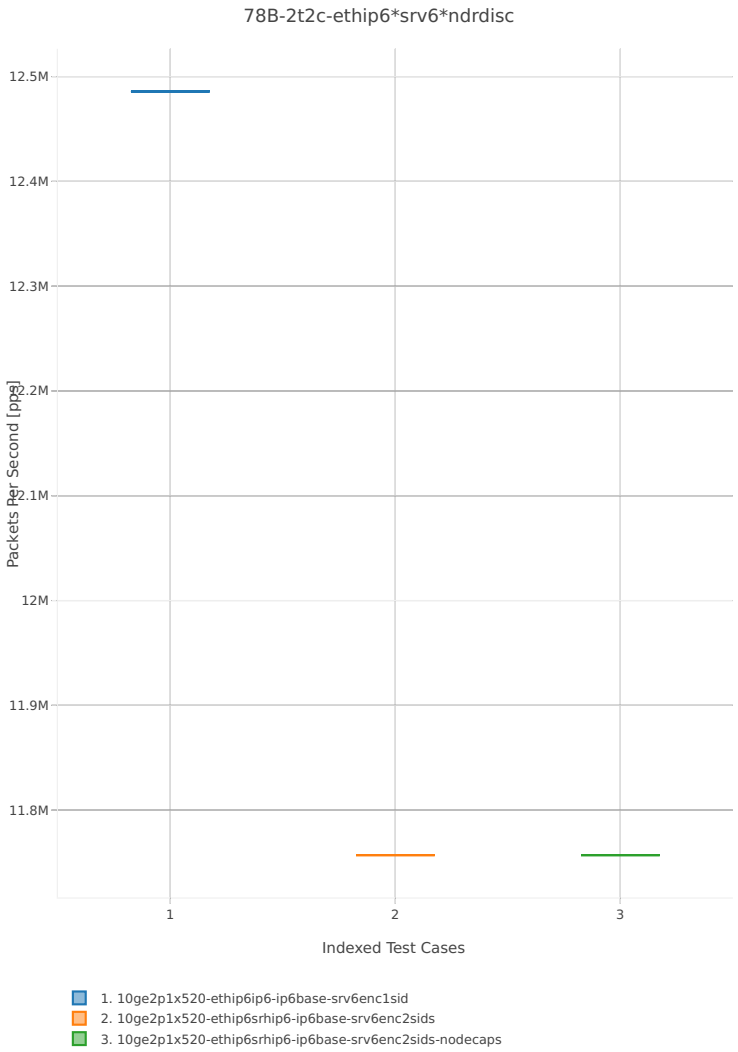


Figure 2. VPP 2threads 2cores - NDR Throughput for Phy-to-Phy SRv6.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[20].

PDR Throughput

VPP PDR 78B packet throughput in 1t1c setup (1thread, 1core) is presented in the graph below. PDR measured for 0.5% packet loss ratio.

[20] https://git.fd.io/csit/tree/tests/vpp/perf/srv6?h=rls1804

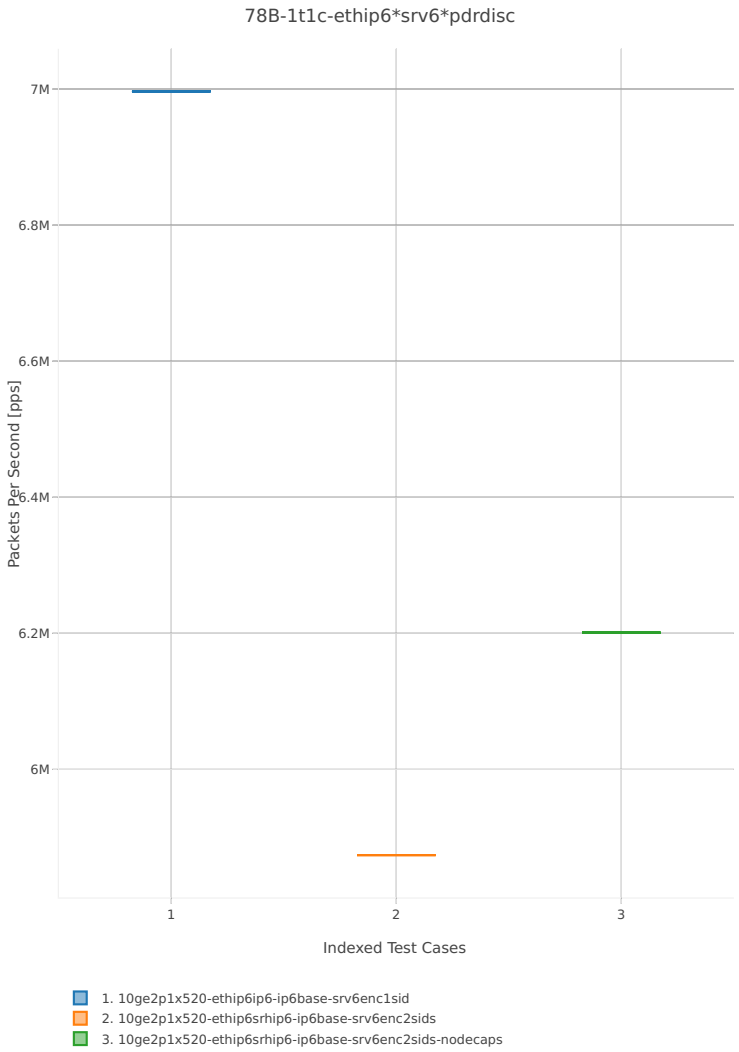


Figure 3. VPP 1thread 1core - PDR Throughput for Phy-to-Phy SRv6.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[21].

VPP PDR 78B packet throughput in 2t2c setup (2thread, 2core) is presented in the graph below. PDR measured for 0.5% packet loss ratio.

[21] https://git.fd.io/csit/tree/tests/vpp/perf/srv6?h=rls1804

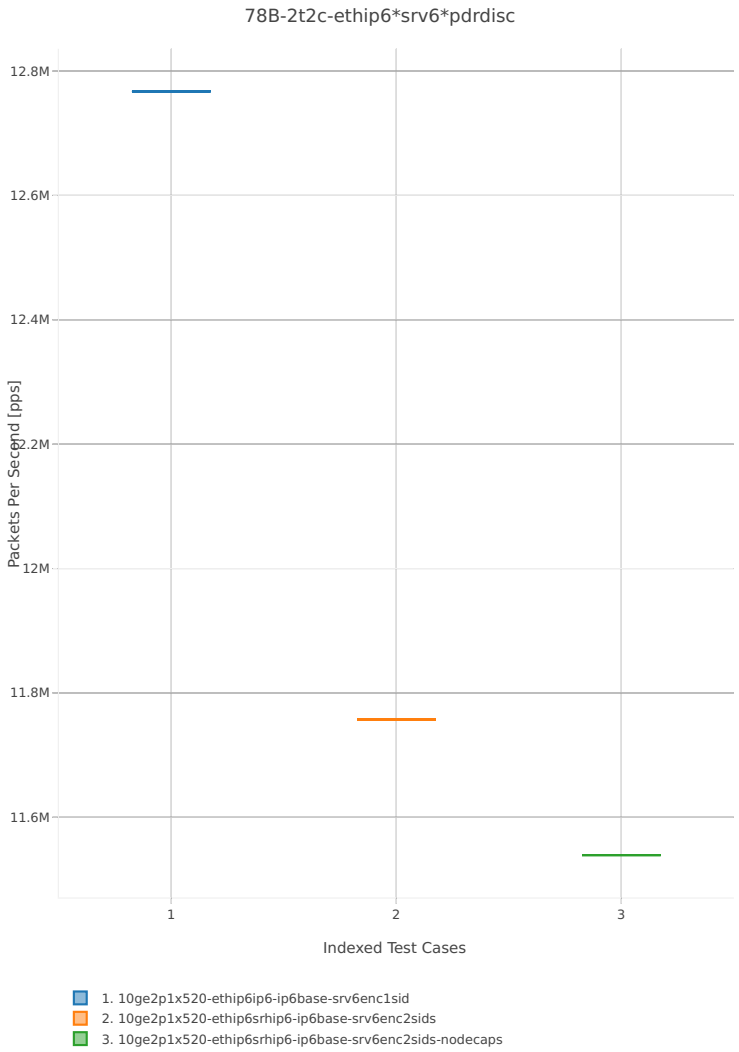


Figure 4. VPP 2thread 2core - PDR Throughput for Phy-to-Phy IPv6 Routed-Forwarding.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[22].

2.3.5 IPv4 Overlay Tunnels

Following sections include summary graphs of VPP Phy-to-Phy performance with IPv4 Overlay Tunnels, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss). Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

NDR Throughput

VPP NDR 64B packet throughput in 1t1c setup (1thread, 1core) is presented in the graph below.

[22] https://git.fd.io/csit/tree/tests/vpp/perf/srv6?h=rls1804

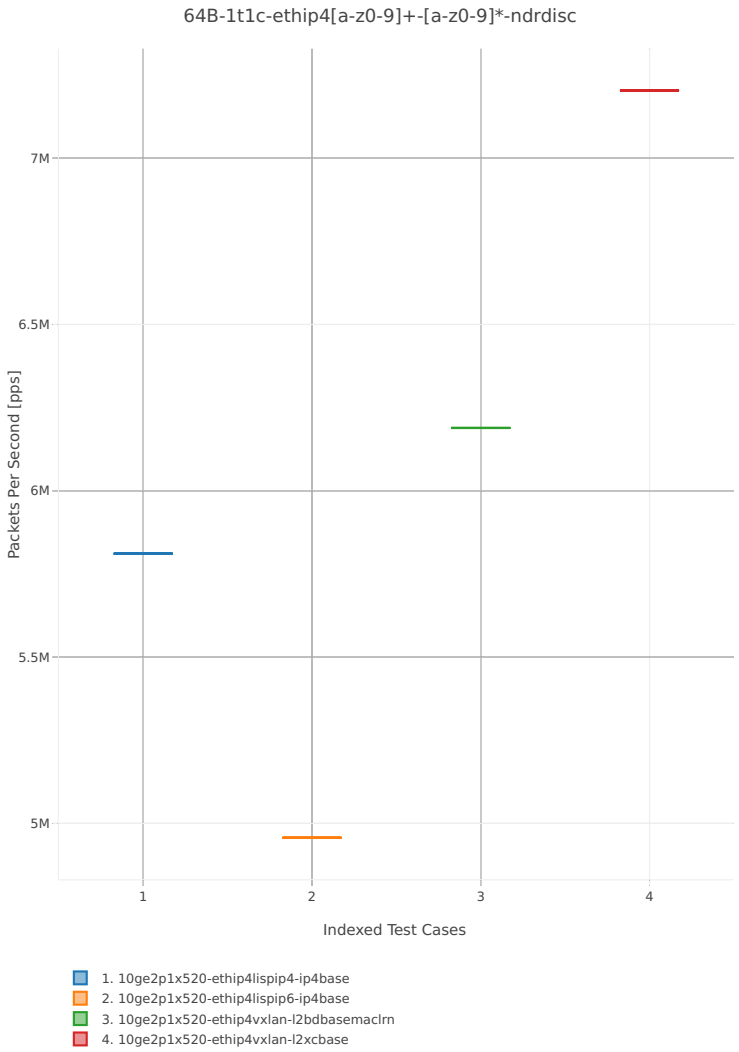


Figure 1. VPP 1thread 1core - NDR Throughput for Phy-to-Phy IPv4 Overlay Tunnels.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/ip4_tunnels
$ grep -E "64B-1t1c-ethip4[a-z0-9]+-[a-z0-9]*-ndrdisc" *
```

VPP NDR 64B packet throughput in 2t2c setup (2thread, 2core) is presented in the graph below.

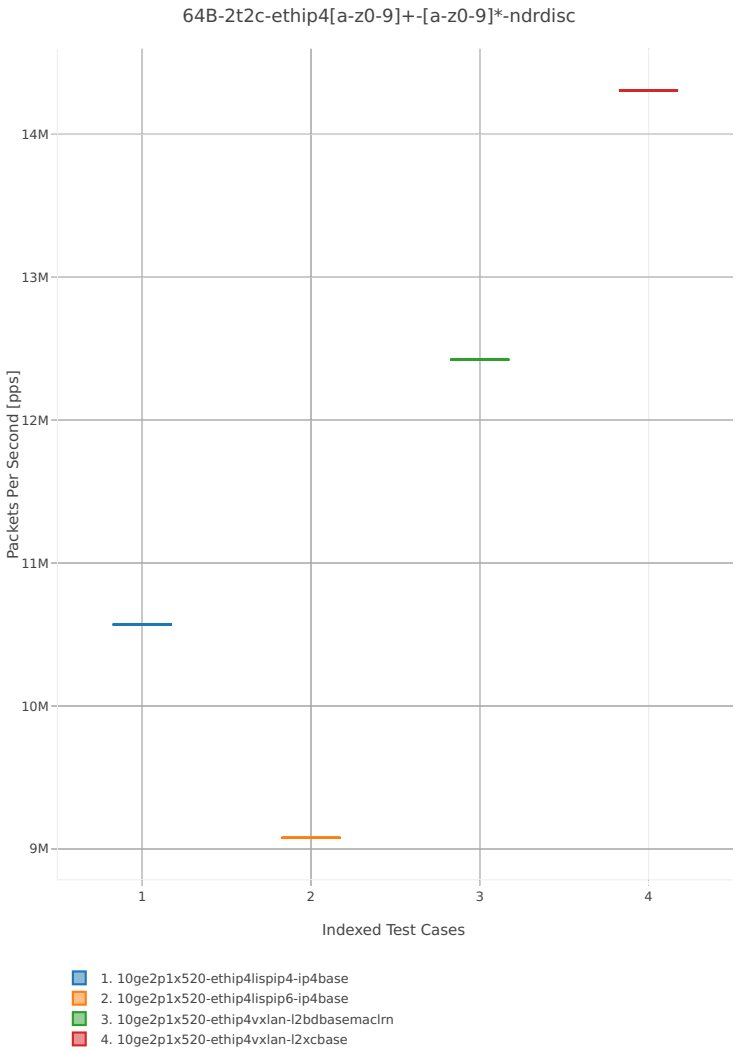


Figure 2. VPP 2threads 2cores - NDR Throughput for Phy-to-Phy IPv4 Overlay Tunnels.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/ip4_tunnels
$ grep -E "64B-2t2c-ethip4[a-z0-9]+-[a-z0-9]*-ndrdisc" *
```

PDR Throughput

VPP PDR 64B packet throughput in 1t1c setup (1thread, 1core) is presented in the graph below. PDR measured for 0.5% packet loss ratio.

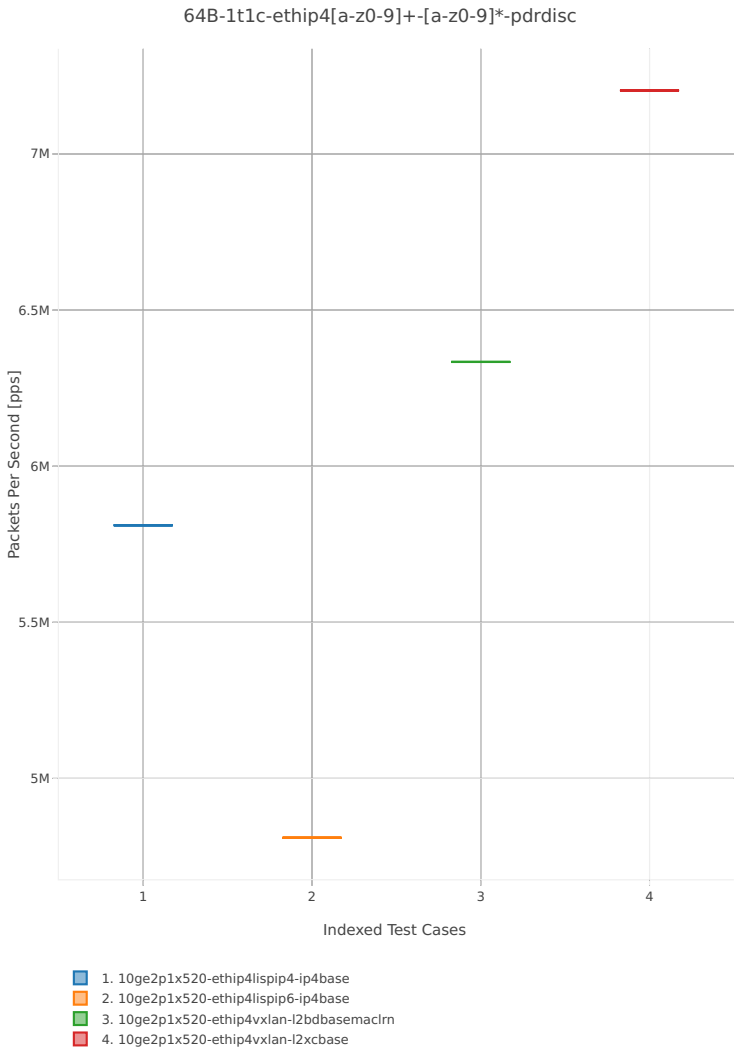


Figure 3. VPP 1thread 1core - PDR Throughput for Phy-to-Phy IPv4 Overlay Tunnels.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/ip4_tunnels
$ grep -E "64B-1t1c-ethip4[a-z0-9]+-[a-z0-9]*-pdrdisc" *
```

VPP PDR 64B packet throughput in 2t2c setup (2thread, 2core) is presented in the graph below. PDR measured for 0.5% packet loss ratio.

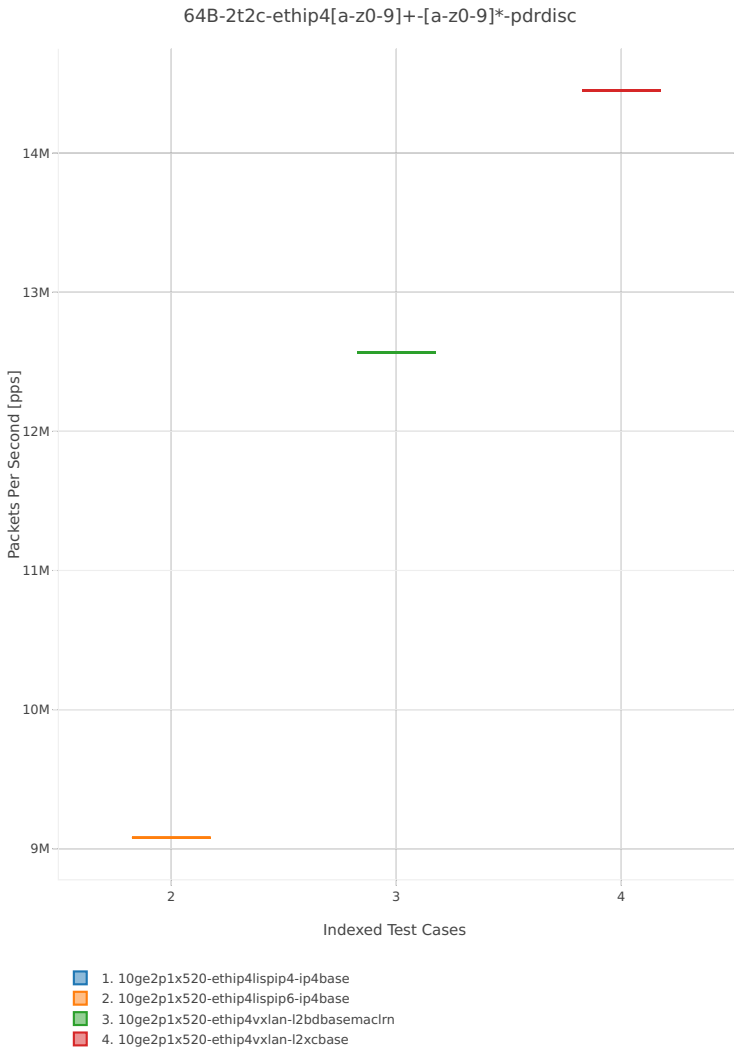


Figure 4. VPP 2thread 2core - PDR Throughput for Phy-to-Phy IPv4 Overlay Tunnels.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/ip4_tunnels
$ grep -E "64B-2t2c-ethip4[a-z0-9]+-[a-z0-9]*-pdrdisc" *
```

2.3.6 IPv6 Overlay Tunnels

Following sections include summary graphs of VPP Phy-to-Phy performance with IPv6 Overlay Tunnels, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss). Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

NDR Throughput

VPP NDR 78B packet throughput in 1t1c setup (1thread, 1core) is presented in the graph below.

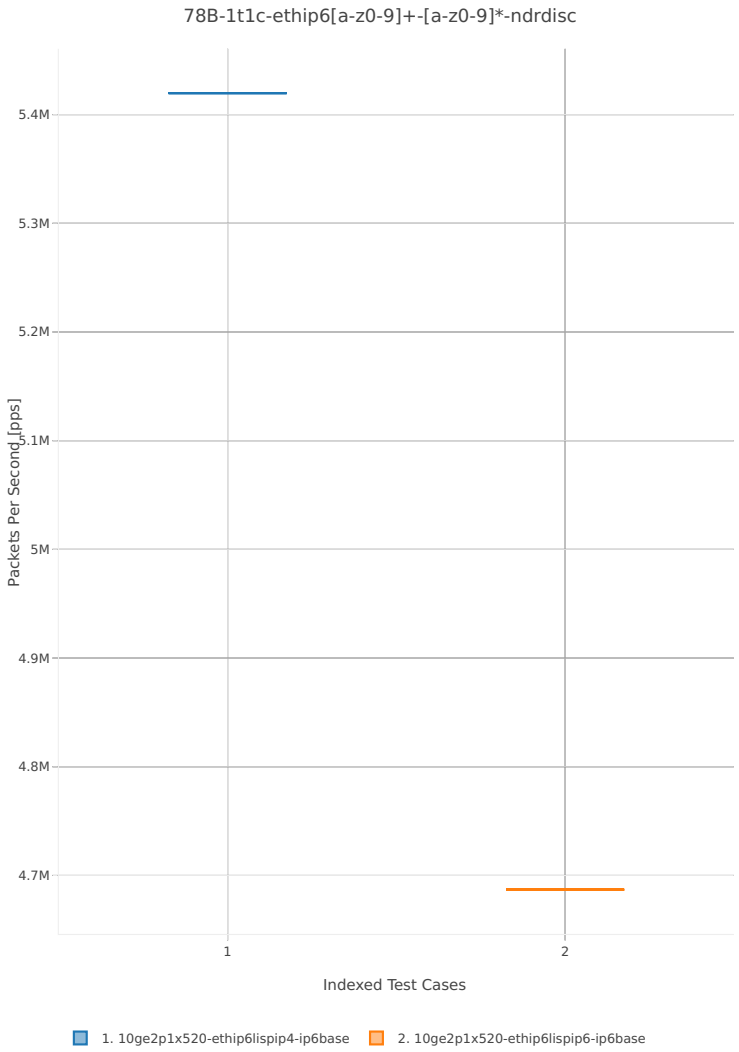


Figure 1. VPP 1thread 1core - NDR Throughput for Phy-to-Phy IPv6 Overlay Tunnels.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/ip6_tunnels
$ grep -E "78B-1t1c-ethip6[a-z0-9]+-[a-z0-9]*-ndrdisc" *
```

VPP NDR 78B packet throughput in 2t2c setup (2thread, 2core) is presented in the graph below.

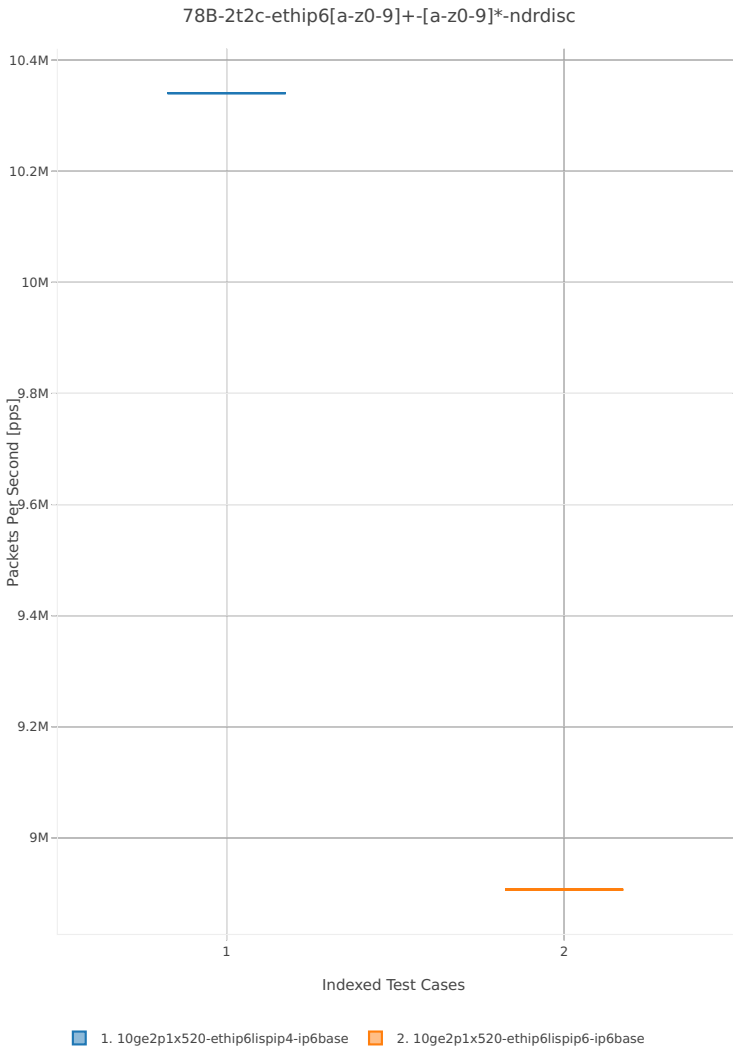


Figure 2. VPP 2threads 2cores - NDR Throughput for Phy-to-Phy IPv6 Overlay Tunnels.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/ip6_tunnels
$ grep -E "78B-1t1c-ethip6[a-z0-9]+-[a-z0-9]*-ndrdisc" *
```

PDR Throughput

VPP PDR 78B packet throughput in 1t1c setup (1thread, 1core) is presented in the graph below. PDR measured for 0.5% packet loss ratio.

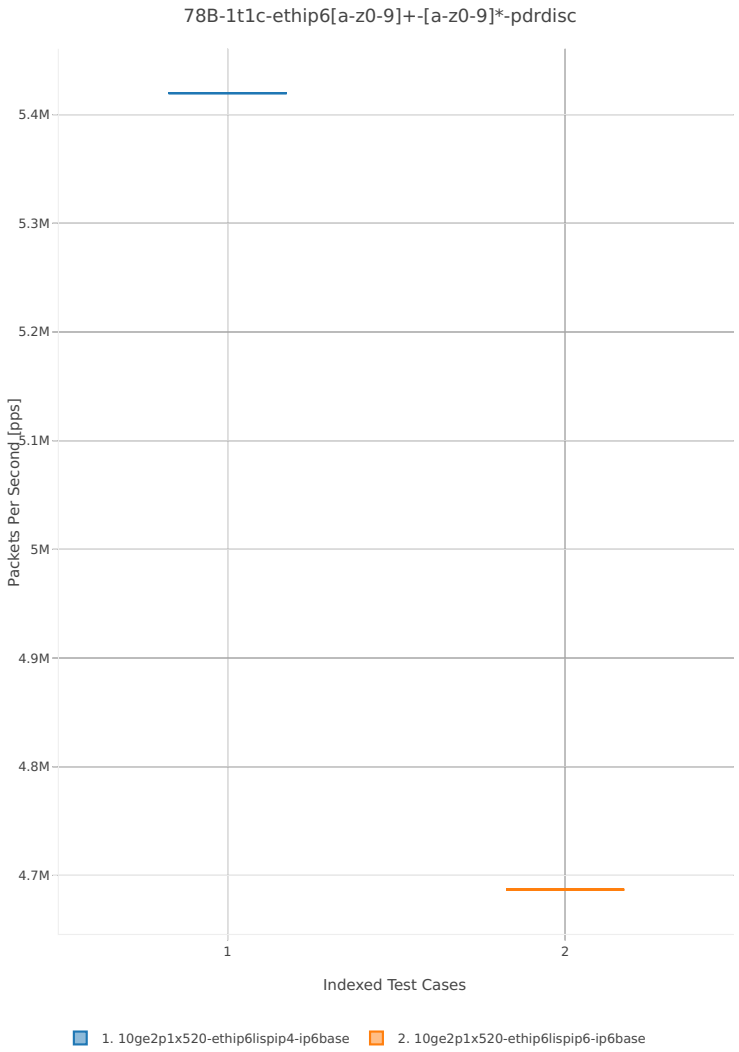


Figure 3. VPP 1thread 1core - PDR Throughput for Phy-to-Phy IPv6 Overlay Tunnels.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/ip6_tunnels
$ grep -E "78B-1t1c-ethip6[a-z0-9]+-[a-z0-9]*-pdrdisc" *
```

VPP PDR 78B packet throughput in 2t2c setup (2thread, 2core) is presented in the graph below. PDR measured for 0.5% packet loss ratio.

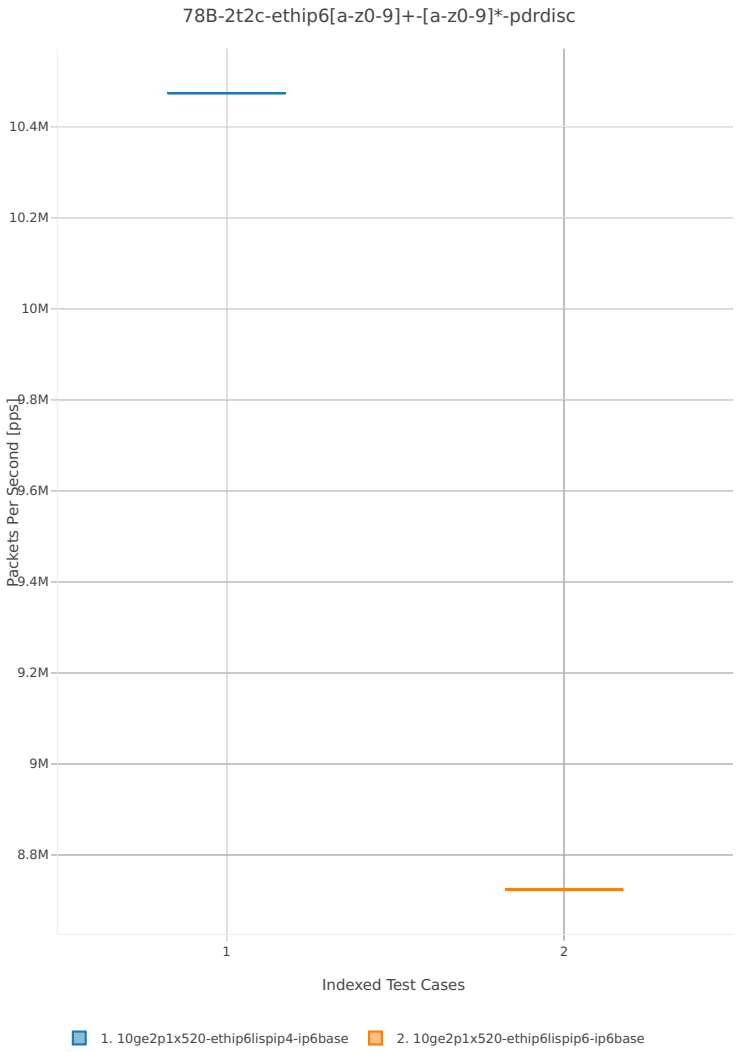


Figure 4. VPP 2thread 2core - PDR Throughput for Phy-to-Phy IPv6 Overlay Tunnels.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/ip6_tunnels
$ grep -E "78B-2t2c-ethip6[a-z0-9]+-[a-z0-9]*-pdrdisc" *
```

2.3.7 VM vhost Connections

Following sections include summary graphs of VPP Phy-to-VM(s)-to-Phy performance with VM virtio and VPP vhost-user virtual interfaces, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss). Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

NDR Throughput

VPP NDR 64B packet throughput in 1t1c setup (1thread, 1core) is presented in the graph below.

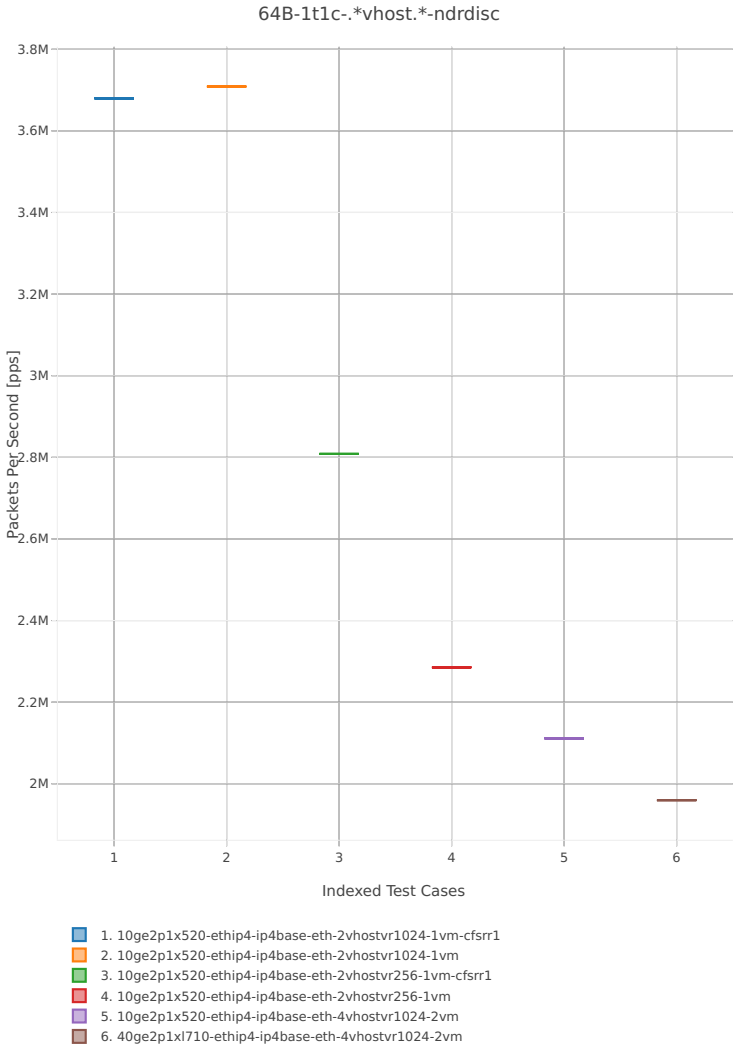


Figure 1a. VPP 1thread 1core - NDR Throughput for Phy-to-VM-to-Phy VM vhost-user selected TCs.

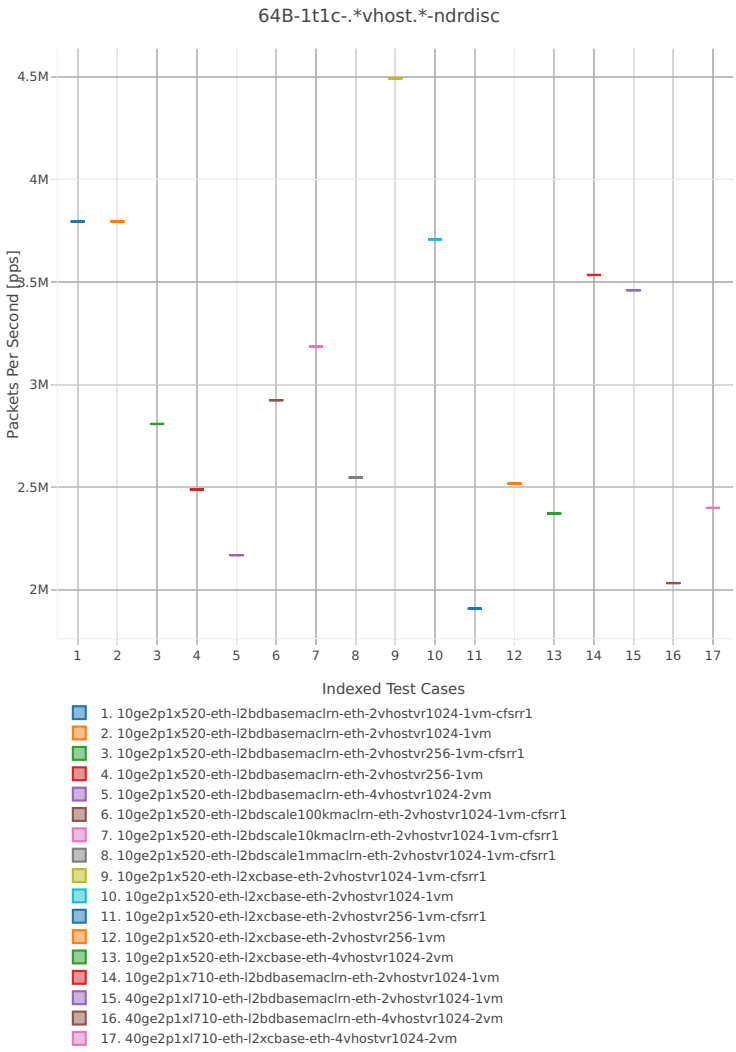


Figure 1b. VPP 1thread 1core - NDR Throughput for Phy-to-VM-to-Phy VM vhost-user selected TCs.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/vm_vhost
$ grep -E "64B-1t1c-.*vhost.*-ndrdisc" *
```

VPP NDR 64B packet throughput in 2t2c setup (2thread, 2core) is presented in the graph below.

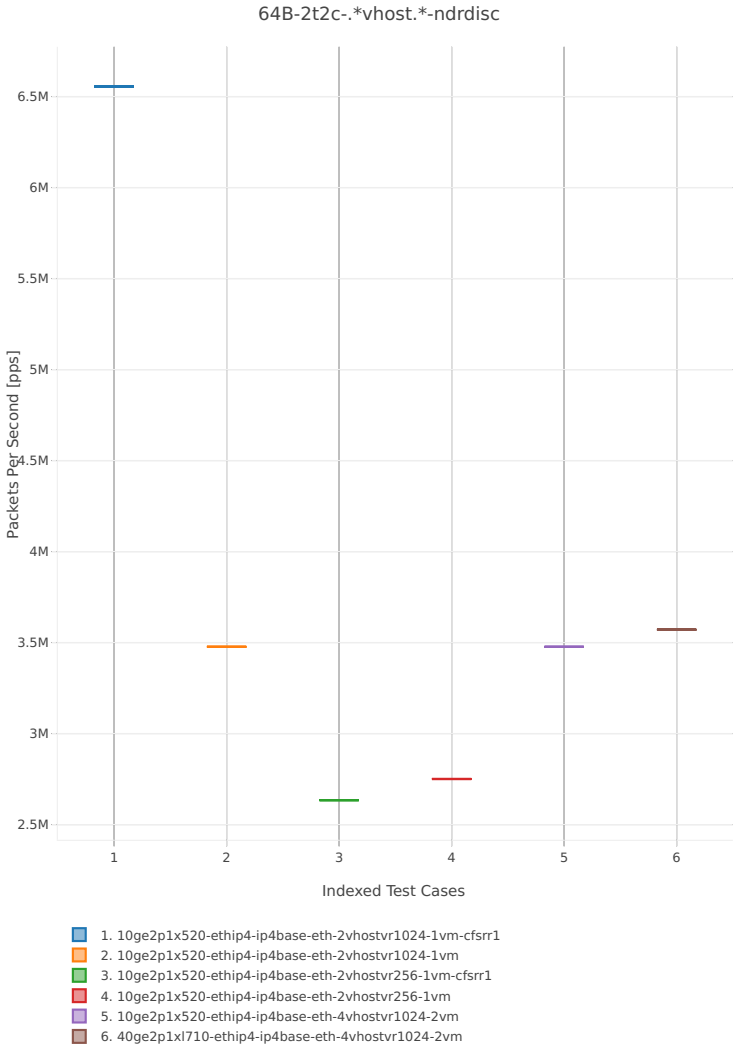


Figure 2a. VPP 2threads 2cores - NDR Throughput for Phy-to-VM-to-Phy VM vhost-user selected TCs.

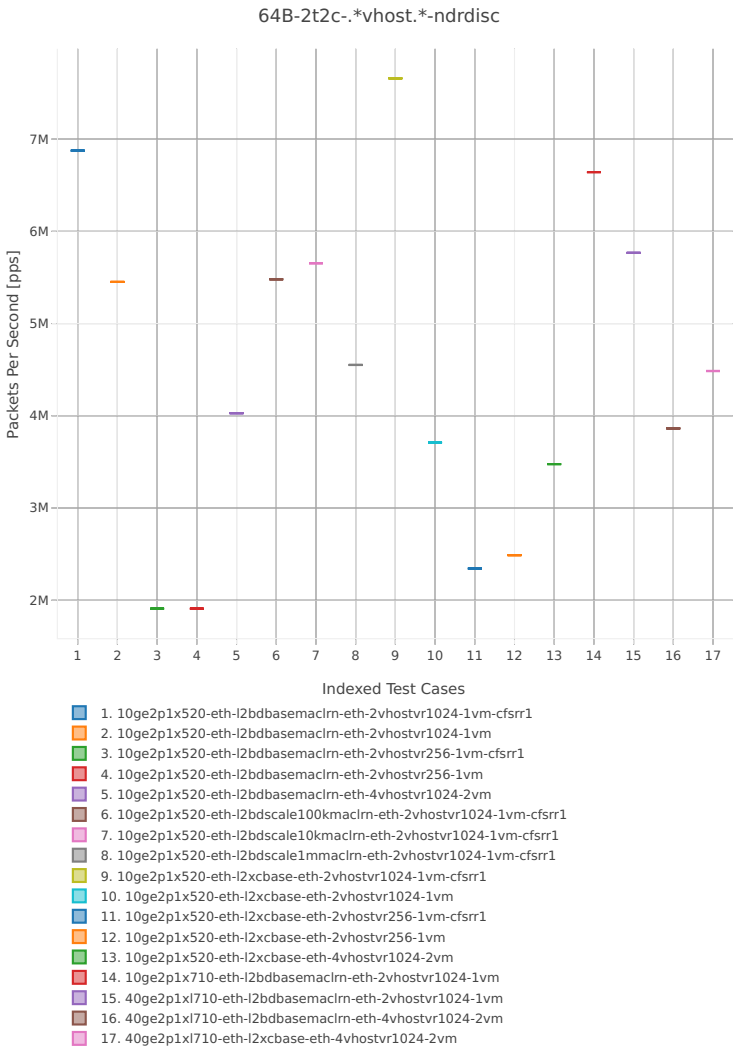


Figure 2b. VPP 2threads 2cores - NDR Throughput for Phy-to-VM-to-Phy VM vhost-user selected TCs.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/vm_vhost
$ grep -E "64B-2t2c-.*vhost.*-ndrdisc" *
```

PDR Throughput

VPP PDR 64B packet throughput in 1t1c setup (1thread, 1core) is presented in the graph below. PDR measured for 0.5% packet loss ratio.

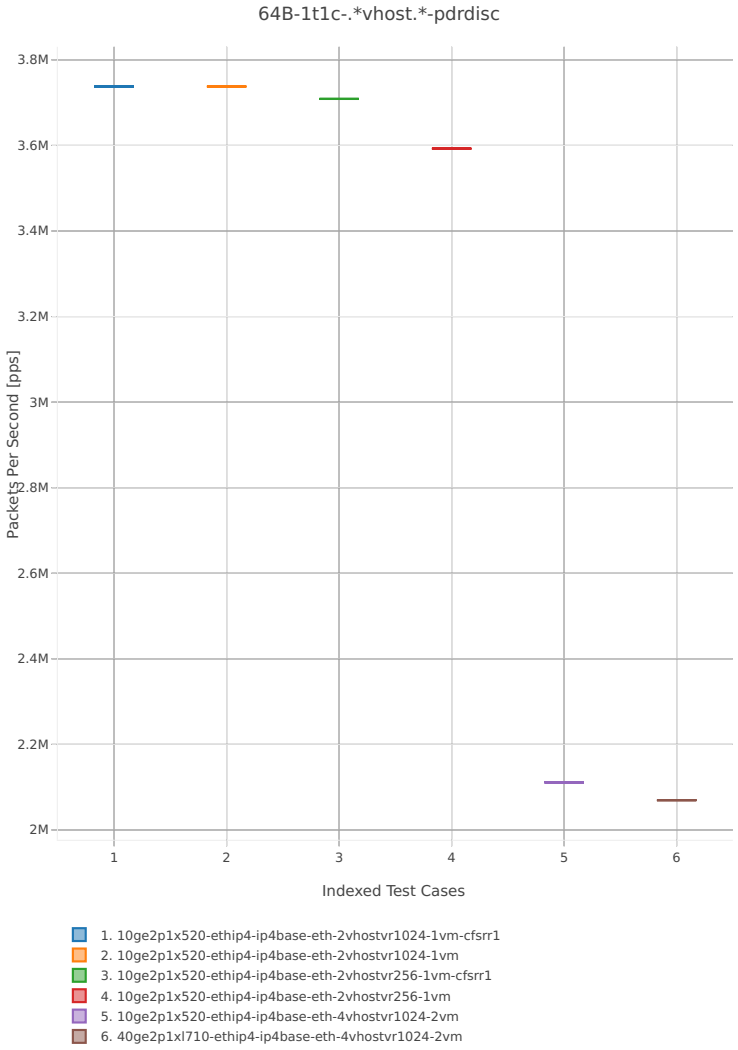


Figure 3a. VPP 1thread 1core - PDR Throughput for Phy-to-VM-to-Phy VM vhost-user selected TCs.

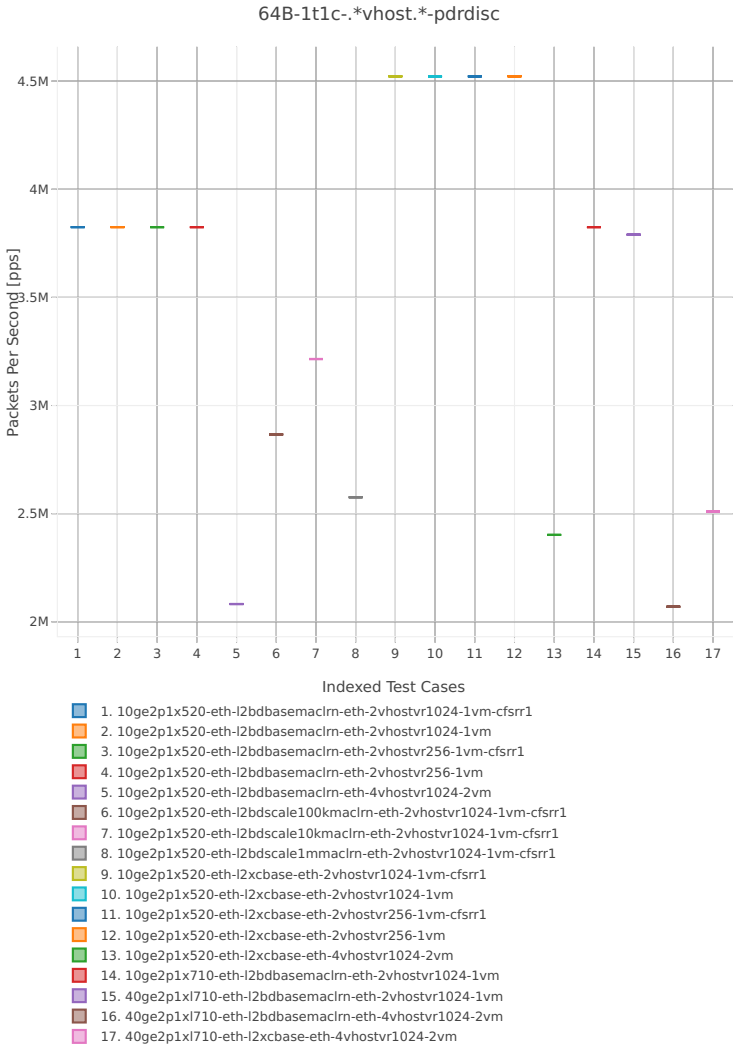


Figure 3b. VPP 1thread 1core - PDR Throughput for Phy-to-VM-to-Phy VM vhost-user selected TCs.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/vm_vhost
$ grep -E "64B-1t1c-.*vhost.*-pdrdisc" *
```

VPP PDR 64B packet throughput in 2t2c setup (2thread, 2core) is presented in the graph below. PDR measured for 0.5% packet loss ratio.

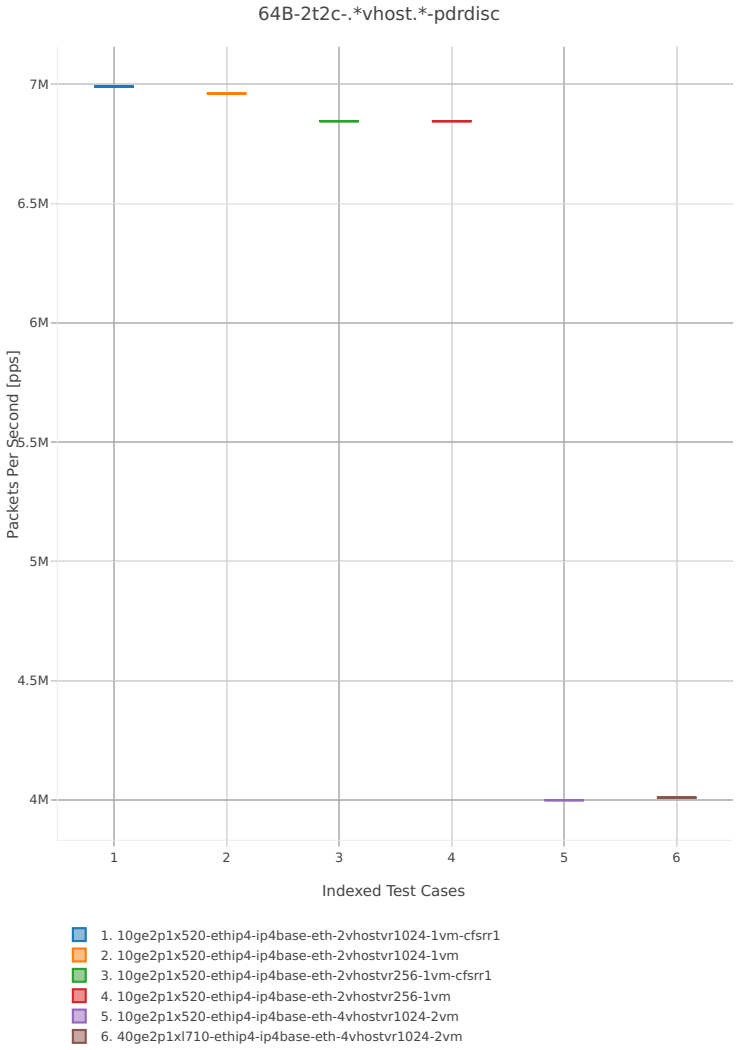


Figure 4a. VPP 2thread 2core - PDR Throughput for Phy-to-VM-to-Phy VM vhost-user selected TCs.

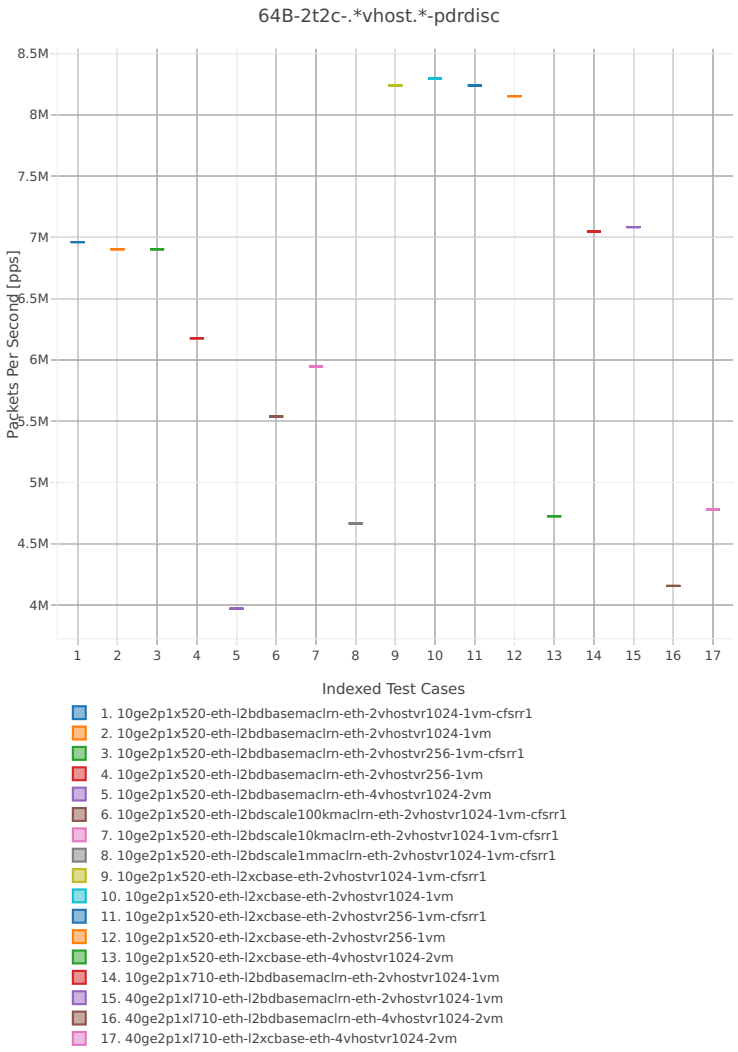


Figure 4b. VPP 2thread 2core - PDR Throughput for Phy-to-VM-to-Phy VM vhost-user selected TCs.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/vm_vhost
$ grep -E "64B-2t2c-.*vhost.*-pdrdisc" *
```

2.3.8 Container memif Connections

Following sections include summary graphs of VPP Phy-to-Phy performance with Container memif Connections, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss). Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

NDR Throughput

VPP NDR 64B packet throughput in 1t1c setup (1thread, 1core) is presented in the graph below.

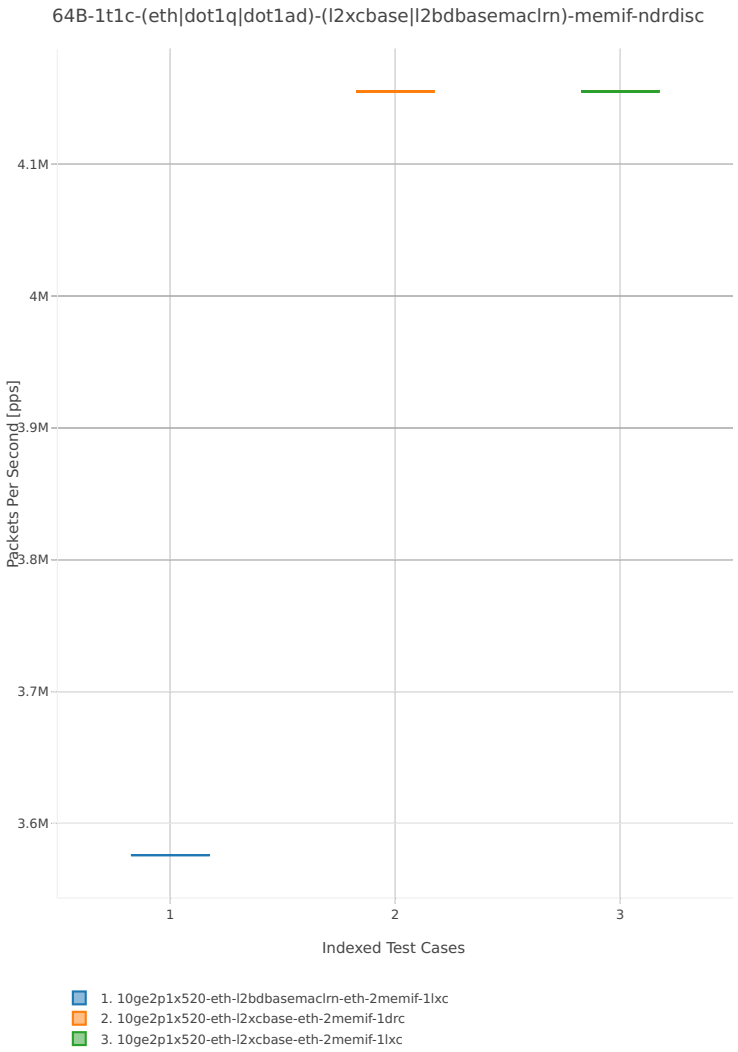


Figure 1. VPP 1thread 1core - NDR Throughput for Phy-to-Phy L2 Ethernet Switching (base).

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/container_memif
$ grep -E "64B-1t1c-(eth|dot1q|dot1ad)-(l2xcbase|l2bdbasemaclrn)-.*ndrdisc" *
```

VPP NDR 64B packet throughput in 2t2c setup (2thread, 2core) is presented in the graph below.

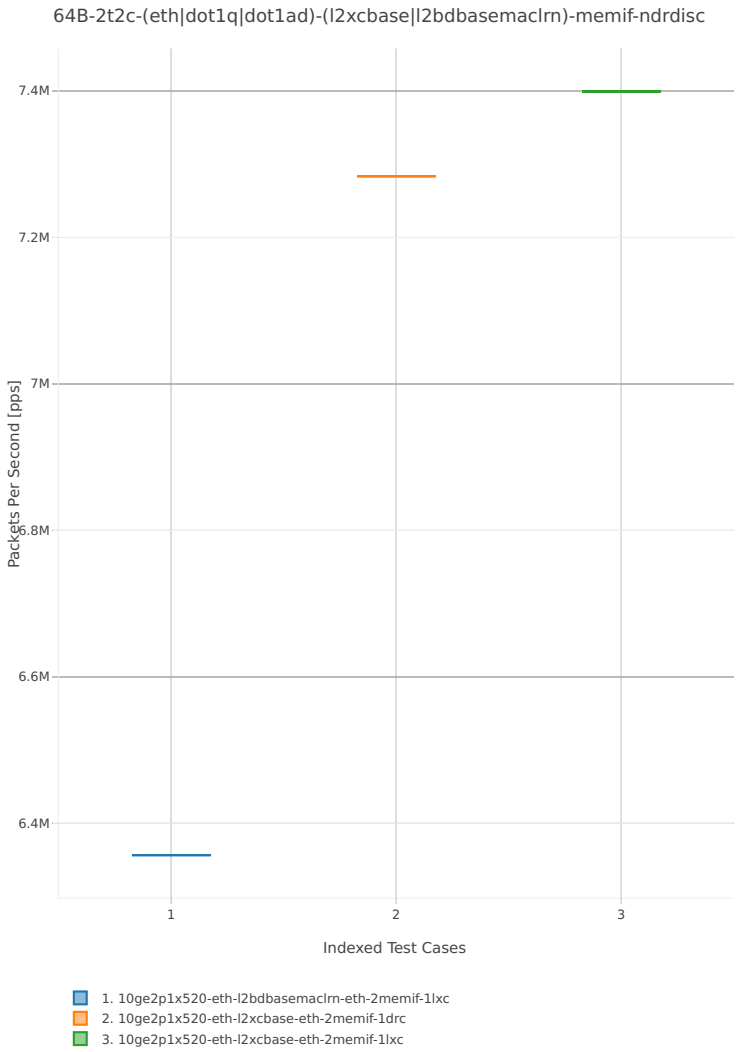


Figure 2. VPP 2threads 2cores - NDR Throughput for Phy-to-Phy L2 Ethernet Switching (base).

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/container_memif
$ grep -E "64B-2t2c-(eth|dot1q|dot1ad)-(l2xcbase|l2bdbasemaclrn)-.*ndrdisc" *
```

PDR Throughput

VPP PDR 64B packet throughput in 1t1c setup (1thread, 1core) is presented in the graph below. PDR measured for 0.5% packet loss ratio.

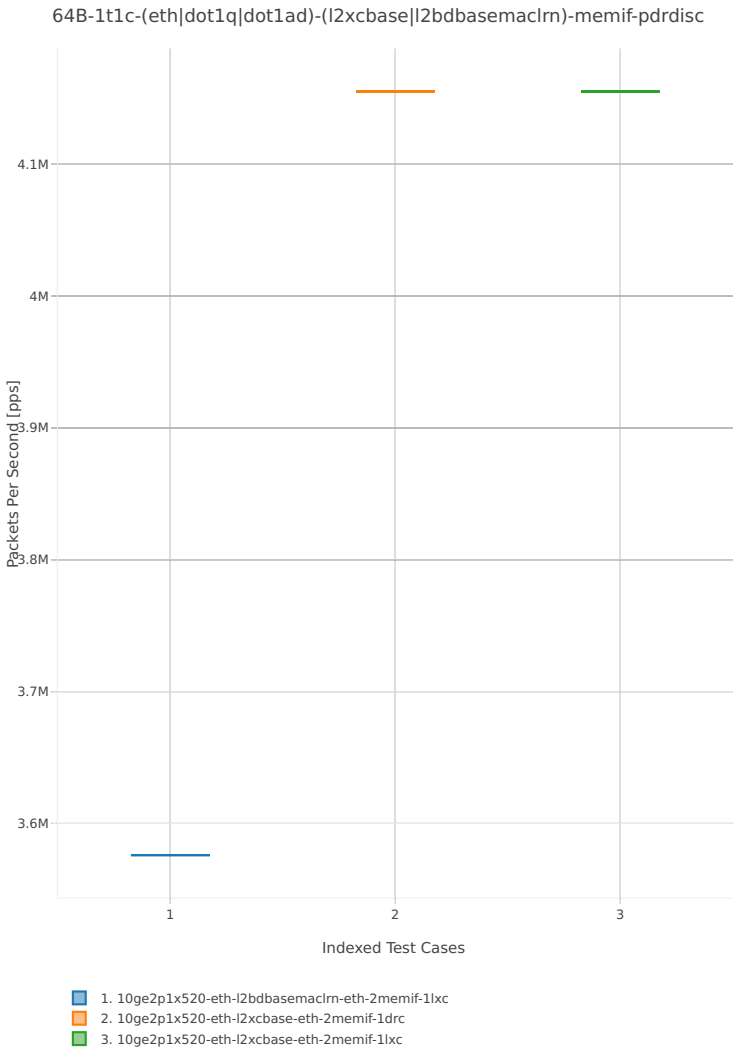


Figure 3. VPP 1thread 1core - PDR Throughput for Phy-to-Phy L2 Ethernet Switching (base).

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/container_memif
$ grep -E "64B-1t1c-(eth|dot1q|dot1ad)-(l2xcbase|l2bdbasemaclrn)-.*pdrdisc" *
```

VPP PDR 64B packet throughput in 2t2c setup (2thread, 2core) is presented in the graph below. PDR measured for 0.5% packet loss ratio.

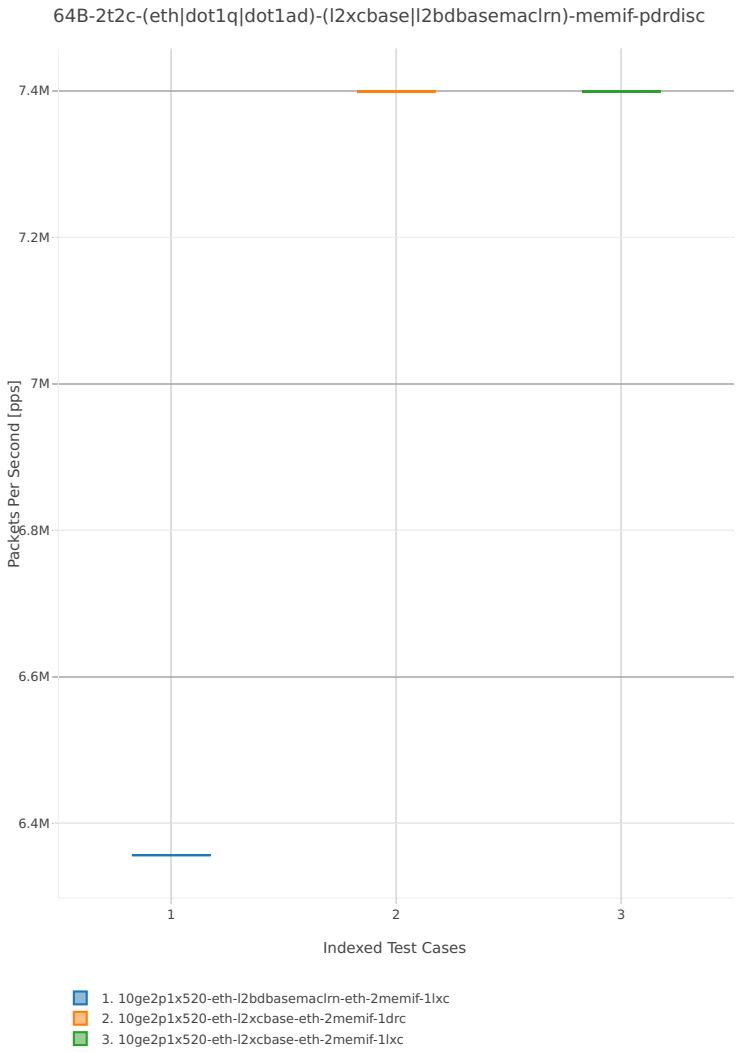


Figure 4. VPP 2thread 2core - PDR Throughput for Phy-to-Phy L2 Ethernet Switching (base).

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/container_memif
$ grep -E "64B-2t2c-(eth|dot1q|dot1ad)-(l2xcbase|l2bdbasemaclrn)-.*pdrdisc" *
```

2.3.9 Container Orchestrated Topologies

Following sections include summary graphs of VPP Phy-to-Phy performance with Container Orchestrated Topologies, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss). Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

NDR Throughput

VPP NDR 64B packet throughput in 1t1c setup (1thread, 1core) is presented in the graph below.

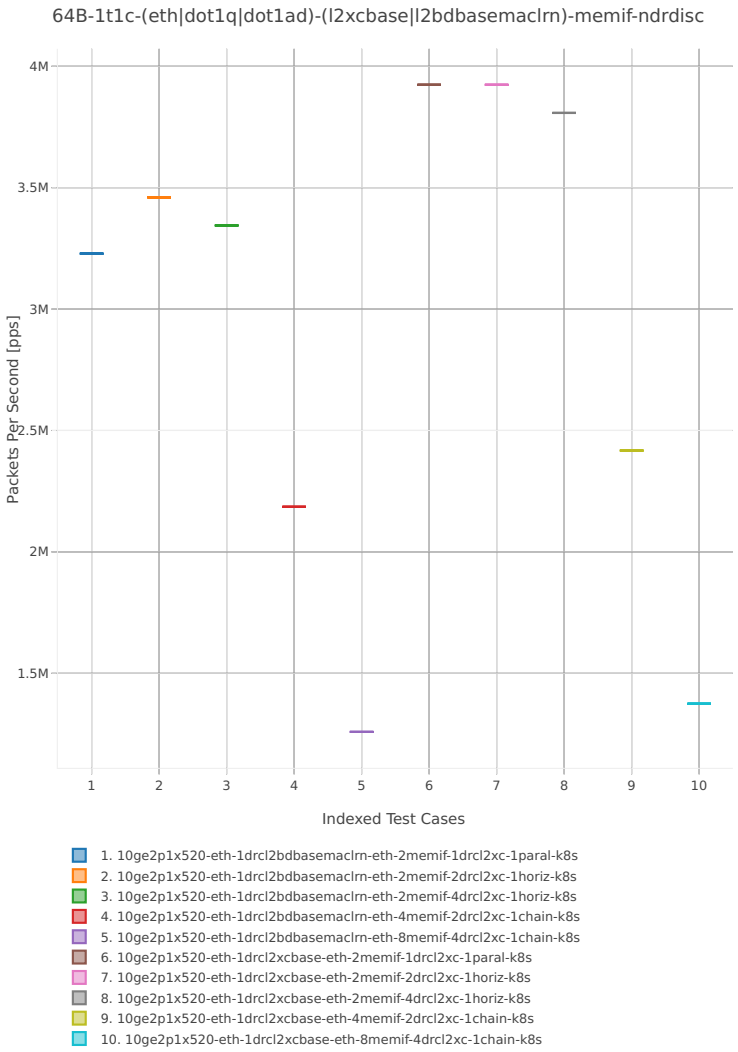


Figure 1. VPP 1thread 1core - NDR Throughput for Phy-to-Phy L2 Ethernet Switching (base).

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/kubernetes/perf/container_memif
$ grep -E "64B-1t1c-(eth|dot1q|dot1ad)-[1-9]drc(l2xcbase|l2bdbasemaclrn)-.*ndrdisc" *
```

VPP NDR 64B packet throughput in 2t2c setup (2thread, 2core) is presented in the graph below.

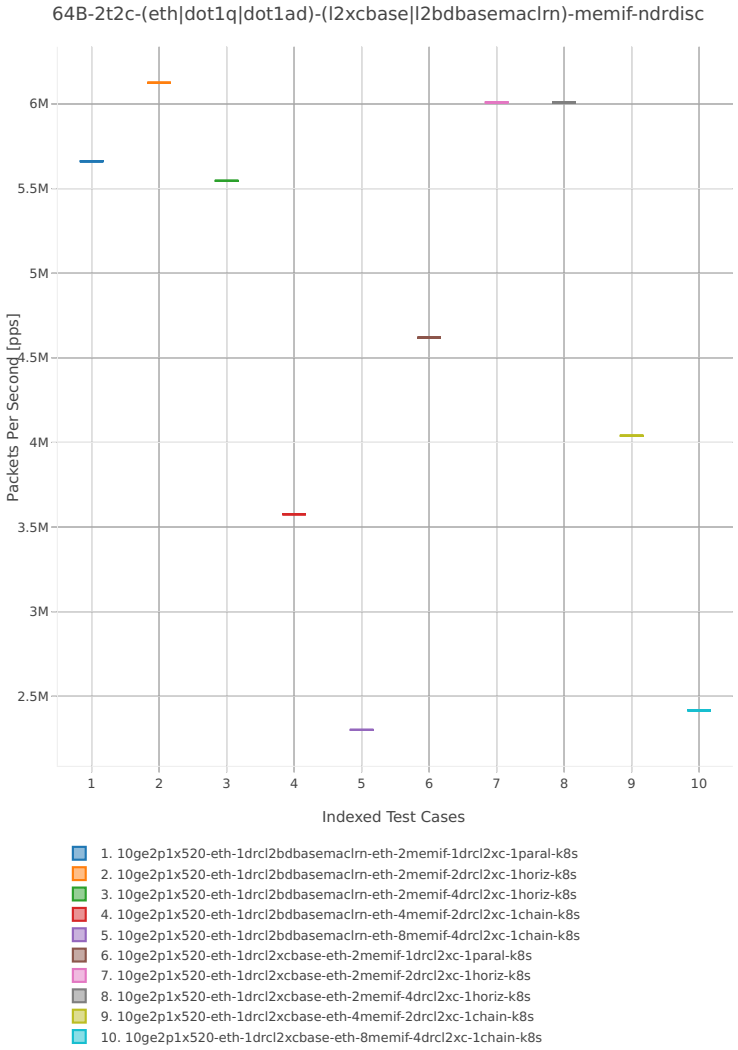


Figure 2. VPP 2threads 2cores - NDR Throughput for Phy-to-Phy L2 Ethernet Switching (base).

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/kubernetes/perf/container_memif
$ grep -E "64B-2t2c-(eth|dot1q|dot1ad)-[1-9]drc(l2xcbase|l2bdbasemaclrn)-.*ndrdisc" *
```

PDR Throughput

VPP PDR 64B packet throughput in 1t1c setup (1thread, 1core) is presented in the graph below. PDR measured for 0.5% packet loss ratio.

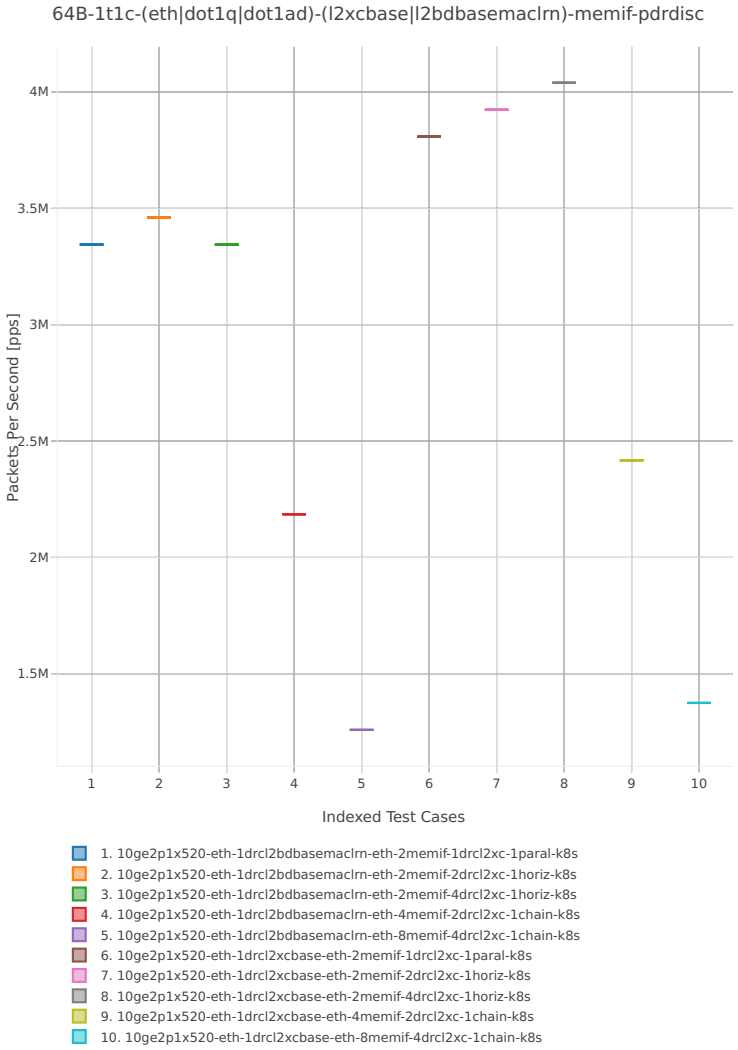


Figure 3. VPP 1thread 1core - PDR Throughput for Phy-to-Phy L2 Ethernet Switching (base).

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/kubernetes/perf/container_memif
$ grep -E "64B-1t1c-(eth|dot1q|dot1ad)-[1-9]drc(l2xcbase|l2bdbasemaclrn)-.*pdrdisc" *
```

VPP PDR 64B packet throughput in 2t2c setup (2thread, 2core) is presented in the graph below. PDR measured for 0.5% packet loss ratio.

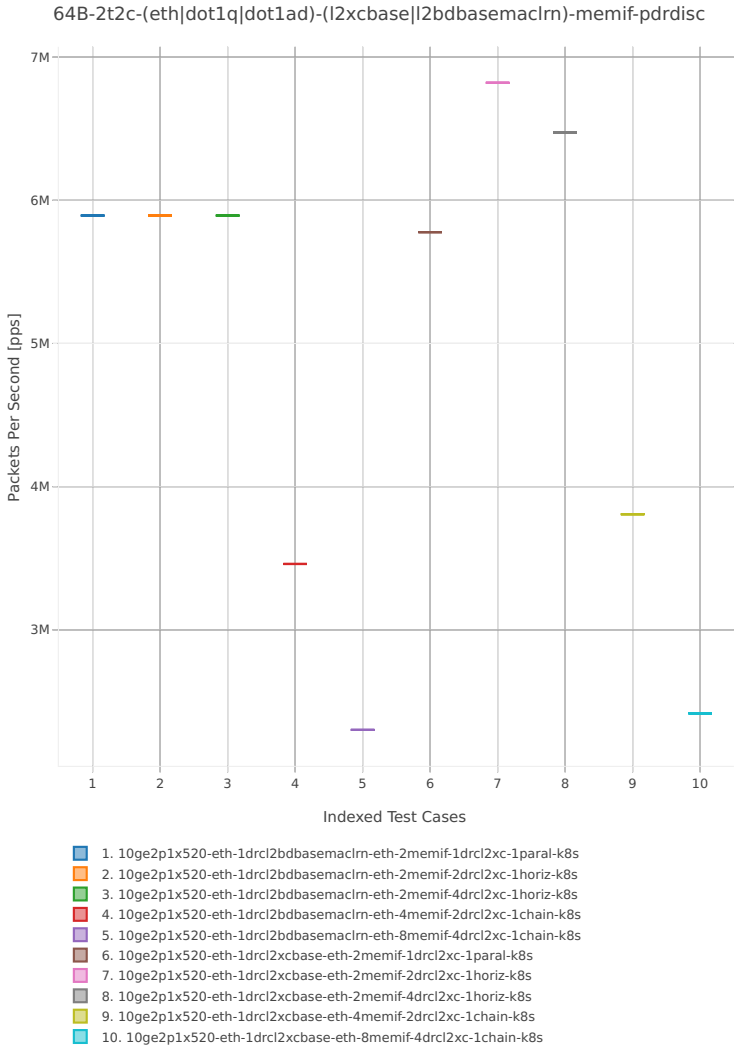


Figure 4. VPP 2thread 2core - PDR Throughput for Phy-to-Phy L2 Ethernet Switching (base).

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/kubernetes/perf/container_memif
$ grep -E "64B-2t2c-(eth|dot1q|dot1ad)-[1-9]drc(l2xcbase|l2bdbasemaclrn)-.*pdrdisc" *
```

2.3.10 IPSec Crypto HW: IP4 Routed-Forwarding

Following sections include summary graphs of VPP Phy-to-Phy performance with IPSec encryption used in combination with IPv4 routed-forwarding, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss). VPP IPSec encryption is accelerated using DPDK cryptodev library driving Intel Quick Assist (QAT) crypto PCIe hardware cards. Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

NDR Throughput

VPP NDR 64B packet throughput in 1t1c setup (1thread, 1core) is presented in the graph below.

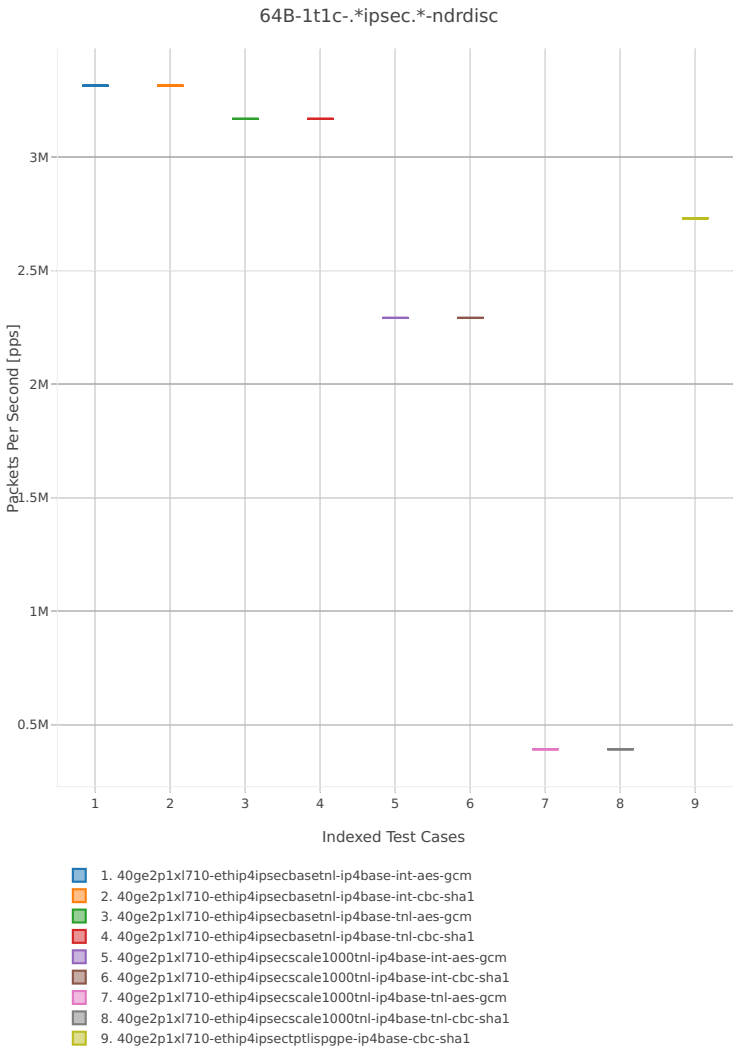


Figure 1. VPP 1thread 1core - NDR Throughput for Phy-to-Phy IPSEC HW.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/crypto
$ grep -E "64B-1t1c-.*ipsec.*-ndrdisc" *
```

VPP NDR 64B packet throughput in 2t2c setup (2thread, 2core) is presented in the graph below.

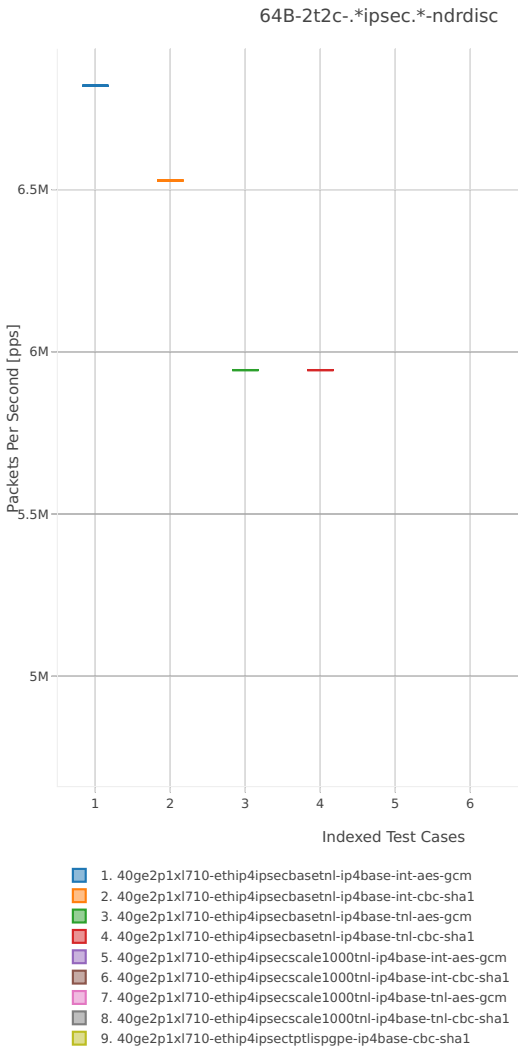


Figure 2. VPP 2threads 2cores - NDR Throughput for Phy-to-Phy IPSEC HW.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/crypto
$ grep -E "64B-2t2c-.*ipsec.*-ndrdisc" *
```

PDR Throughput

VPP PDR 64B packet throughput in 1t1c setup (1thread, 1core) is presented in the graph below. PDR measured for 0.5% packet loss ratio.

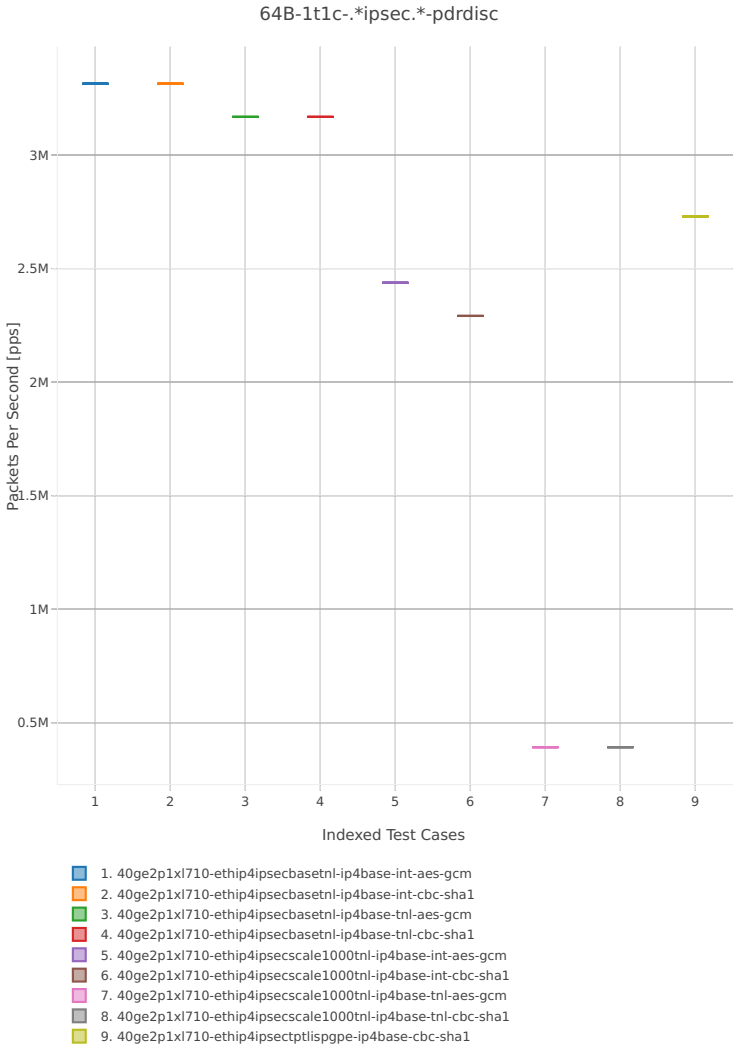


Figure 3. VPP 1thread 1core - PDR Throughput for Phy-to-Phy IPSEC HW.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/crypto
$ grep -E "64B-1t1c-.*ipsec.*-pdrdisc" *
```

VPP PDR 64B packet throughput in 2t2c setup (2thread, 2core) is presented in the graph below. PDR measured for 0.5% packet loss ratio.

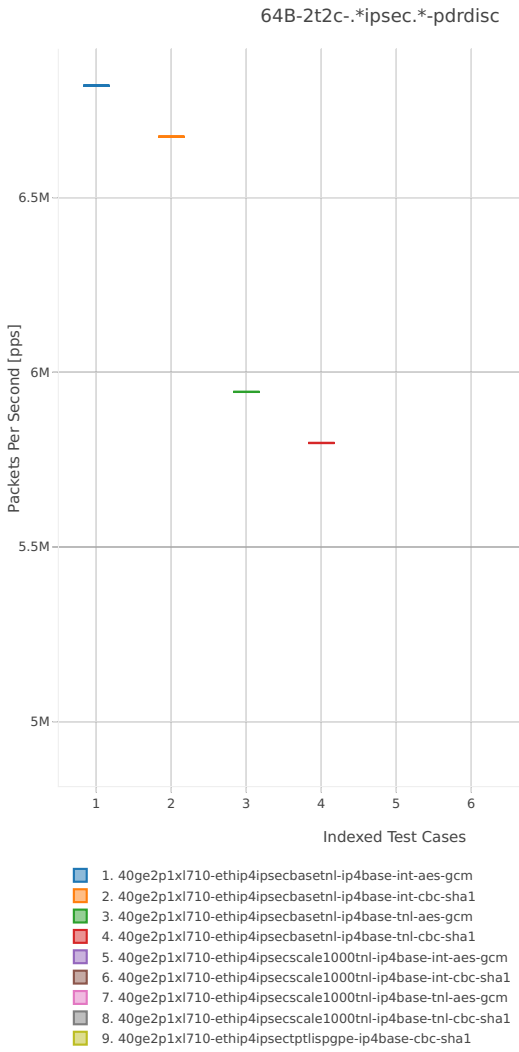


Figure 4. VPP 2thread 2core - PDR Throughput for Phy-to-Phy IPSEC HW.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/crypto
$ grep -E "64B-2t2c-.*ipsec.*-pdrdisc" *
```

2.4 Throughput Speedup Multi-Core

Throughput Speedup Multi-Core graphs are generated based on results from multiple executions of the same CSIT performance tests across the three physical testbeds within LF FD.io labs. Plotted grouped bars illustrate the 64B/78B packet throughput speedup ratio for 2- and 4-core multi- threaded VPP configurations relative to 1-core configurations.

Note: Test results have been generated by FD.io test executor vpp performance jobs[23] with Robot Framework result files csit-vpp-perf-*.zip archived here.

[23] https://jenkins.fd.io/view/csit/job/csit-vpp-perf-1801-all

2.4.1 L2 Ethernet Switching

Following sections include Throughput Speedup Analysis for VPP multi- core multi-thread configurations with no Hyper-Threading, specifically for tested 2t2c (2threads, 2cores) and 4t4c scenarios. 1t1c throughput results are used as a reference for reported speedup ratio. Input data used for the graphs comes from Phy-to-Phy 64B performance tests with VPP L2 Ethernet switching, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss).

NDR Throughput

VPP NDR 64B packet throughput speedup ratio is presented in the graphs below for 10ge2p1x520 and 40ge2p1xl710 network interface cards.

NIC 10ge2p1x520

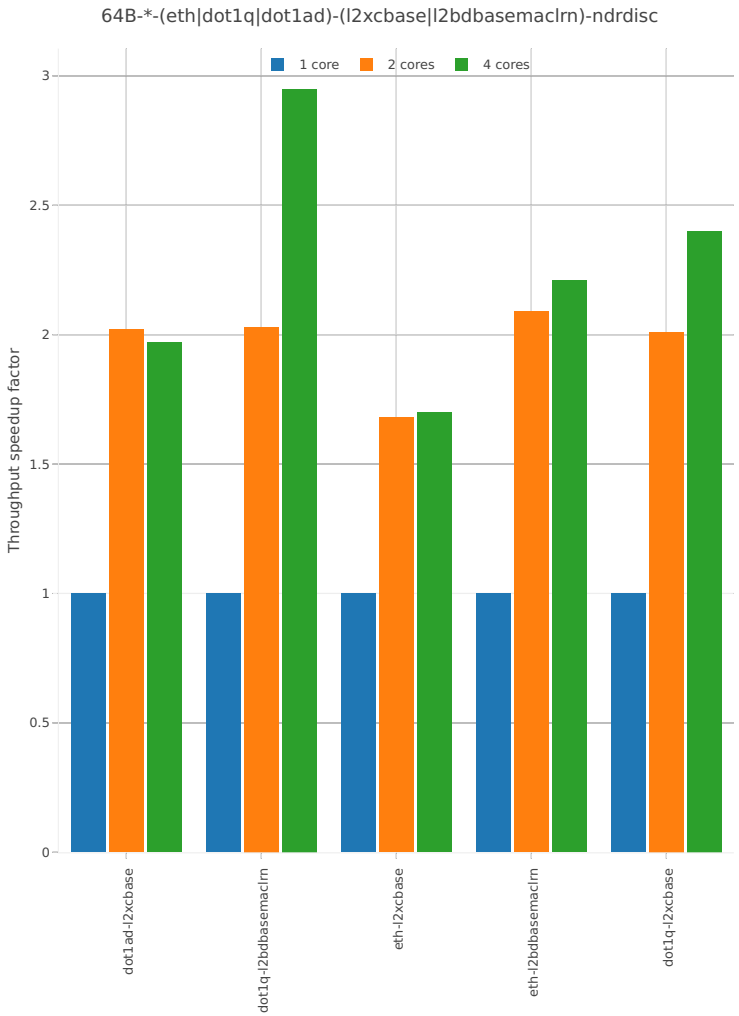


Figure 1. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized NDR Throughput for Phy-to-Phy L2 Ethernet Switching.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[24].

NIC 40ge2p1xl710

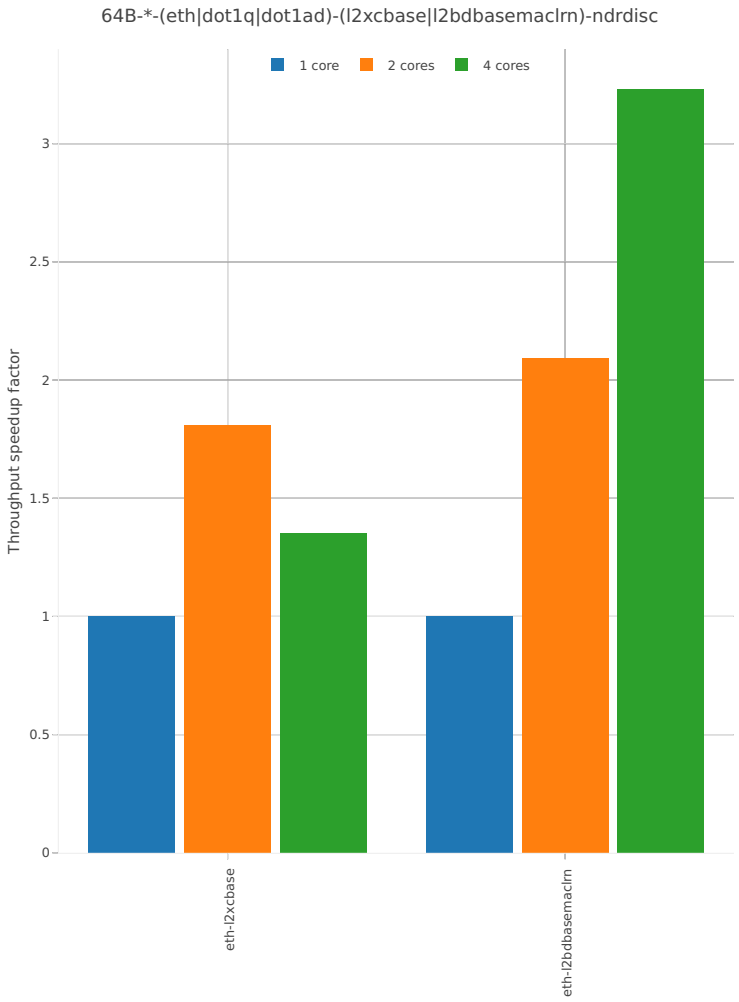


Figure 2. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized NDR Throughput for Phy-to-Phy L2 Ethernet Switching.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[25].

PDR Throughput

VPP PDR 64B packet throughput speedup ratio is presented in the graphs below for 10ge2p1x520 and 40ge2p1xl710 network interface cards. PDR measured for 0.5% packet loss ratio.

[24] https://git.fd.io/csit/tree/tests/vpp/perf/l2?h=rls1804
[25] https://git.fd.io/csit/tree/tests/vpp/perf/l2?h=rls1804

NIC 10ge2p1x520

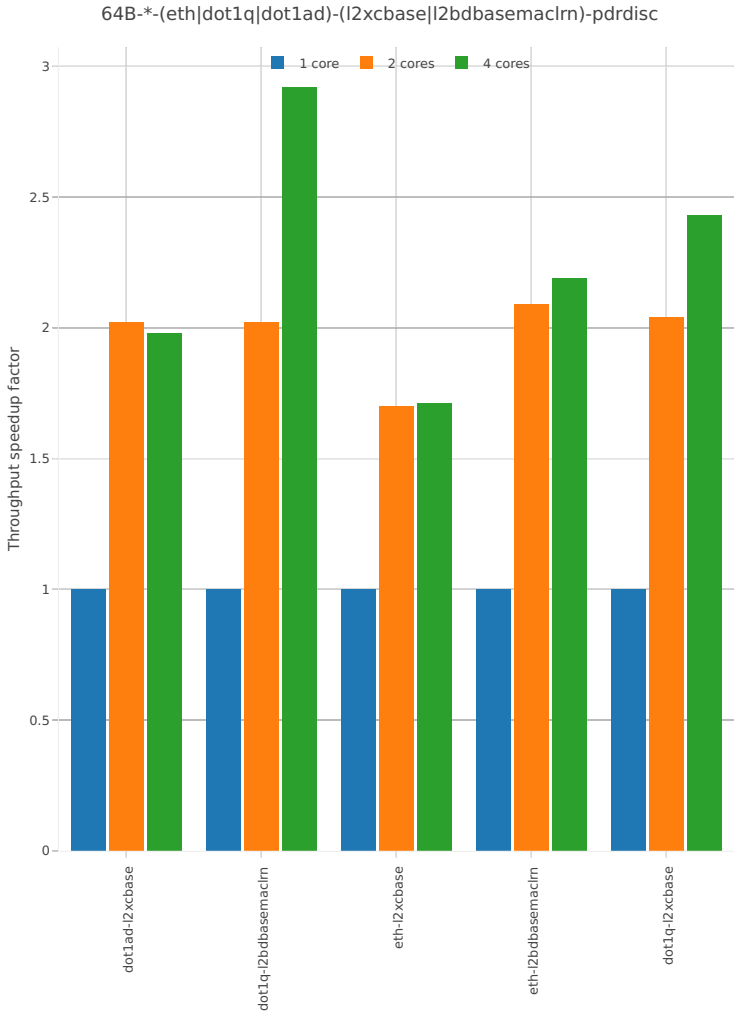


Figure 3. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized PDR Throughput for Phy-to-Phy L2 Ethernet Switching.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[26].

2.4.2 IPv4 Routed-Forwarding

Following sections include Throughput Speedup Analysis for VPP multi- core multi-thread configurations with no Hyper-Threading, specifically for tested 2t2c (2threads, 2cores) and 4t4c scenarios. 1t1c throughput results are used as a reference for reported speedup ratio. Input data used for the graphs comes from Phy-to-Phy 64B performance tests with VPP IPv4 Routed-Forwarding, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss).

[26] https://git.fd.io/csit/tree/tests/vpp/perf/l2?h=rls1804

NDR Throughput

VPP NDR 64B packet throughput speedup ratio is presented in the graphs below for 10ge2p1x520 and 40ge2p1xl710 network interface cards.

NIC 10ge2p1x520

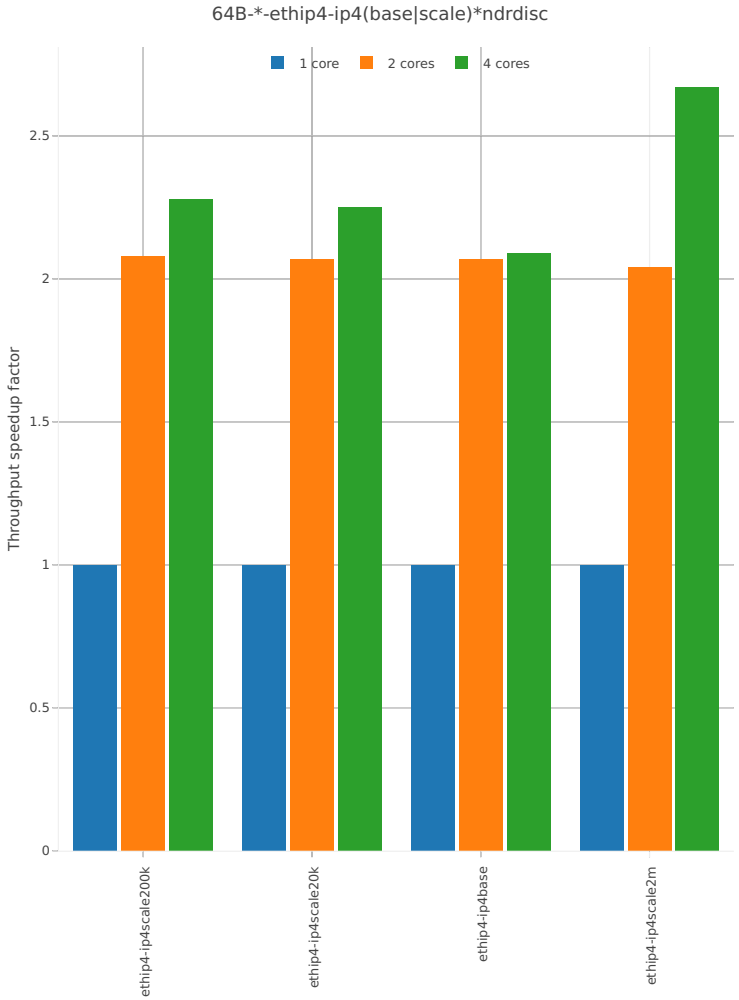


Figure 1. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized NDR Throughput for Phy-to-Phy IPv4 Routed-Forwarding.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[27].

[27] https://git.fd.io/csit/tree/tests/vpp/perf/ip4?h=rls1804

NIC 40ge2p1xl710

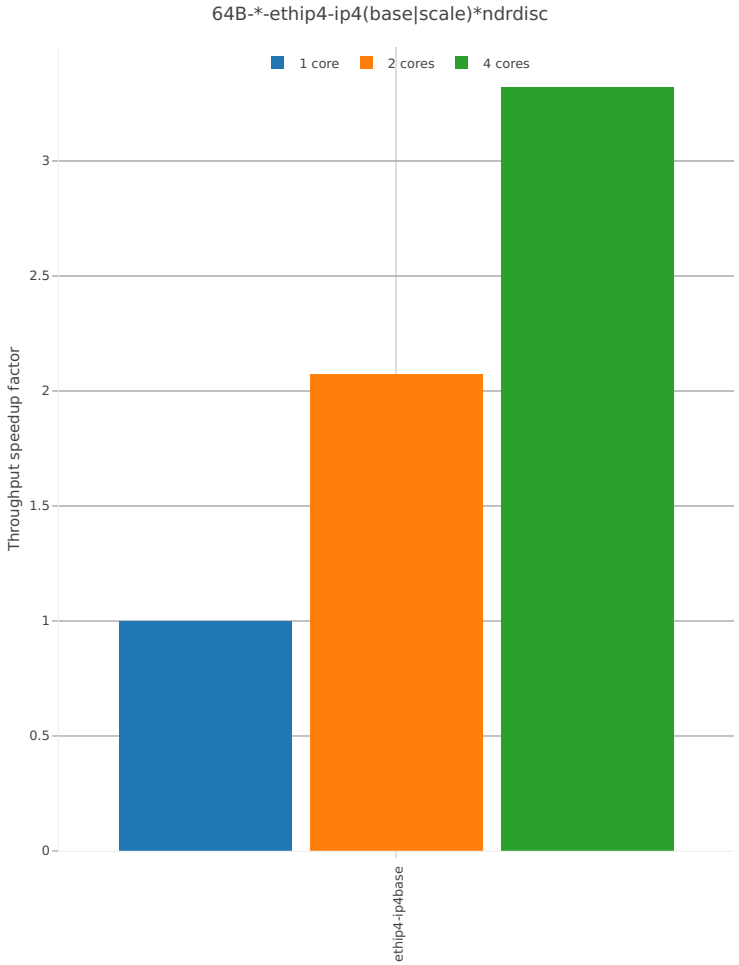


Figure 2. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized NDR Throughput for Phy-to-Phy IPv4 Routed-Forwarding.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[28].

PDR Throughput

VPP PDR 64B packet throughput speedup ratio is presented in the graphs below for 10ge2p1x520 and 40ge2p1xl710 network interface cards. PDR measured for 0.5% packet loss ratio.

[28] https://git.fd.io/csit/tree/tests/vpp/perf/ip4?h=rls1804

NIC 10ge2p1x520

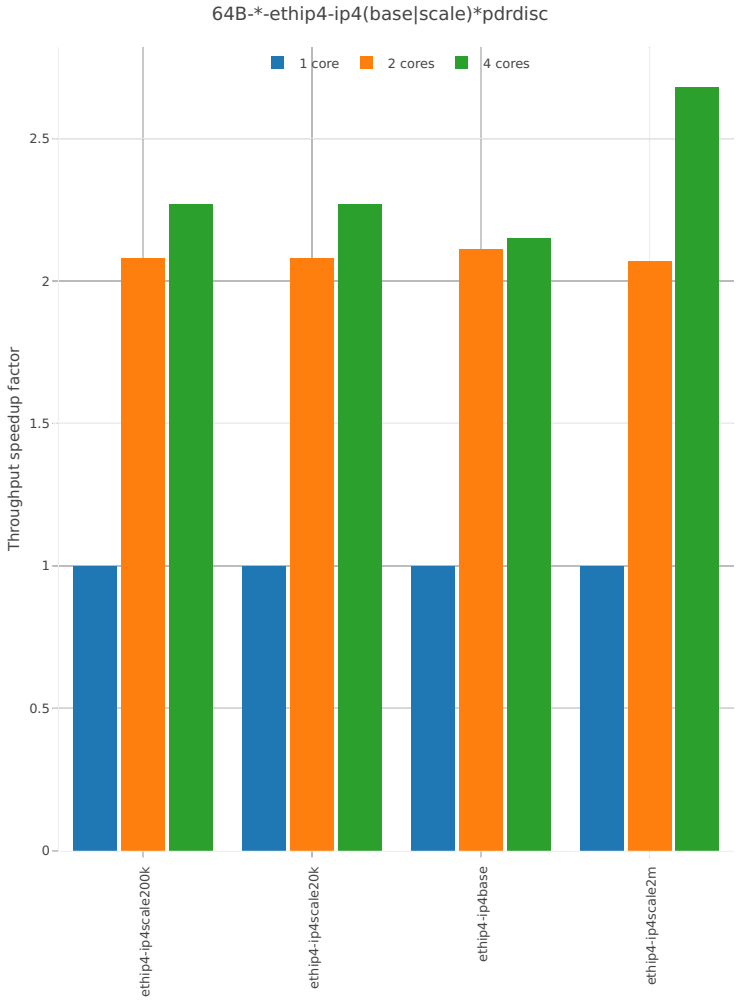


Figure 3. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized PDR Throughput for Phy-to-Phy IPv4 Routed-Forwarding.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[29].

2.4.3 IPv6 Routed-Forwarding

Following sections include Throughput Speedup Analysis for VPP multi- core multi-thread configurations with no Hyper-Threading, specifically for tested 2t2c (2threads, 2cores) and 4t4c scenarios. 1t1c throughput results are used as a reference for reported speedup ratio. Input data used for the graphs comes from Phy-to-Phy 78B performance tests with VPP IPv6 Routed-Forwarding, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss).

[29] https://git.fd.io/csit/tree/tests/vpp/perf/ip4?h=rls1804

NDR Throughput

VPP NDR 78B packet throughput speedup ratio is presented in the graphs below for 10ge2p1x520 and 40ge2p1xl710 network interface cards.

NIC 10ge2p1x520

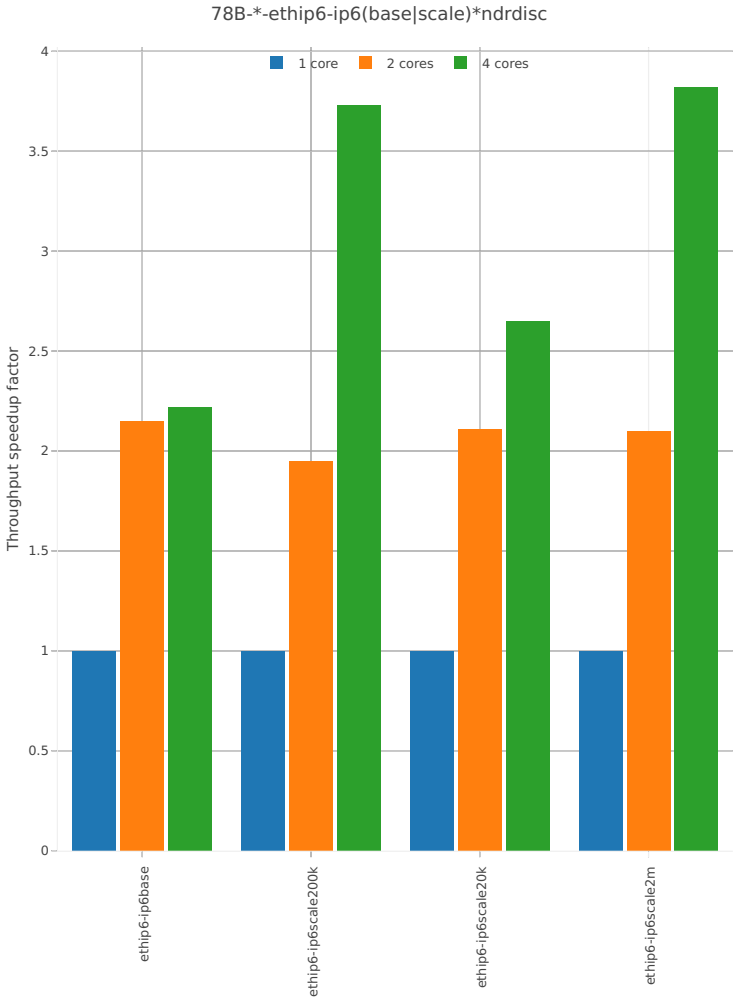


Figure 1. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized NDR Throughput for Phy-to-Phy IPv6 Routed-Forwarding.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[30].

[30] https://git.fd.io/csit/tree/tests/vpp/perf/ip6?h=rls1804

NIC 40ge2p1xl710

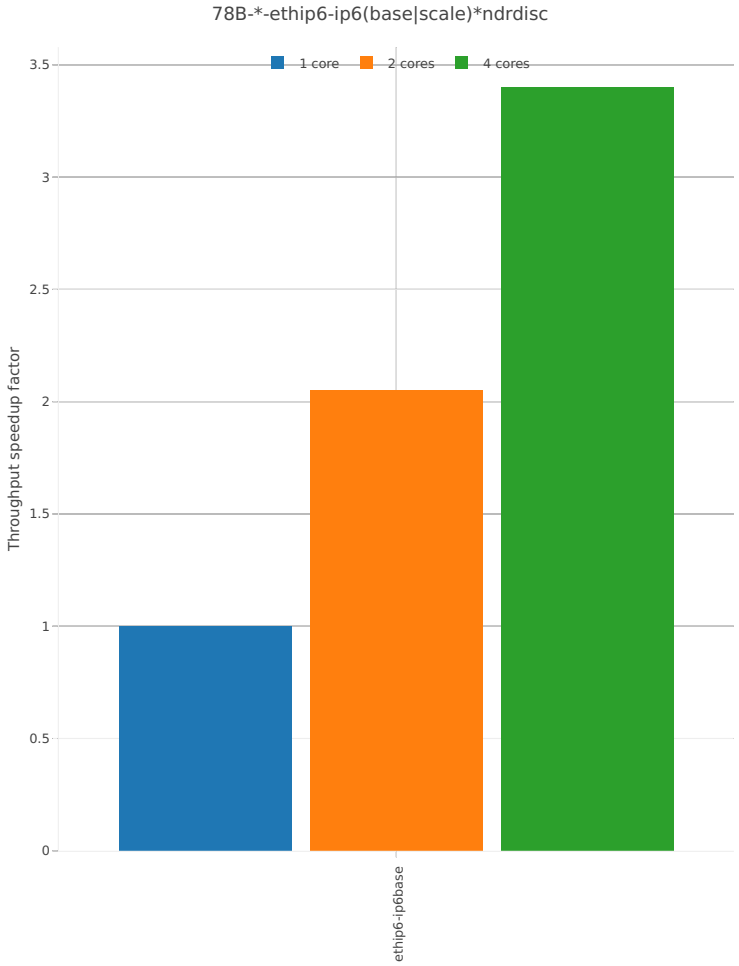


Figure 2. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized NDR Throughput for Phy-to-Phy IPv6 Routed-Forwarding.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[31].

PDR Throughput

VPP PDR 78B packet throughput speedup ratio is presented in the graphs below for 10ge2p1x520 and 40ge2p1xl710 network interface cards. PDR measured for 0.5% packet loss ratio.

[31] https://git.fd.io/csit/tree/tests/vpp/perf/ip6?h=rls1804

NIC 10ge2p1x520

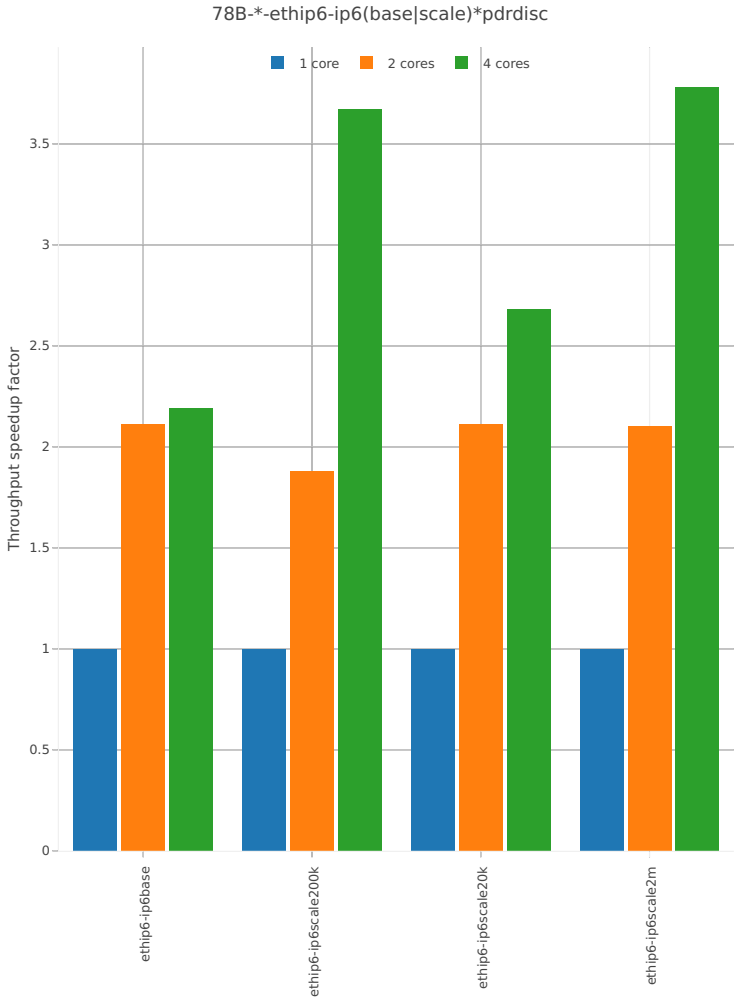


Figure 3. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized PDR Throughput for Phy-to-Phy IPv6 Routed-Forwarding.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[32].

2.4.4 SRv6

Following sections include Throughput Speedup Analysis for VPP multi- core multi-thread configurations with no Hyper-Threading, specifically for tested 2t2c (2threads, 2cores) and 4t4c scenarios. 1t1c throughput results are used as a reference for reported speedup ratio. Input data used for the graphs comes from Phy-to-Phy 78B performance tests with VPP SRv6, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss).

[32] https://git.fd.io/csit/tree/tests/vpp/perf/ip6?h=rls1804

NDR Throughput

VPP NDR 78B packet throughput speedup ratio is presented in the graphs below for 10ge2p1x520 network interface card.

NIC 10ge2p1x520

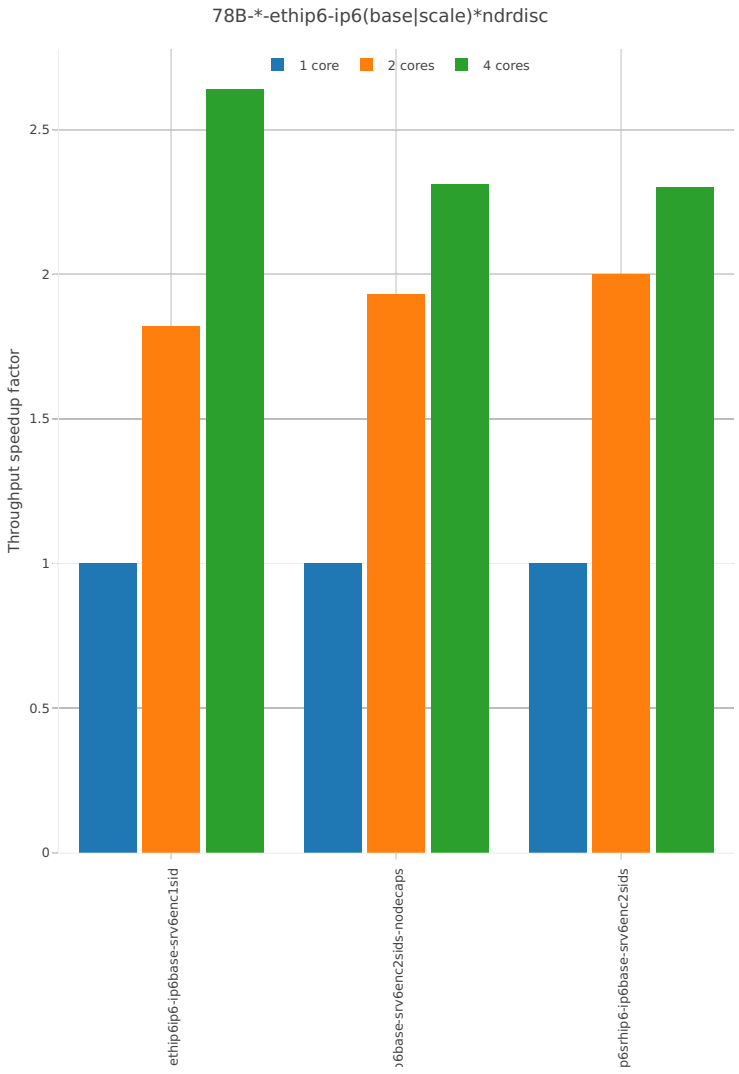


Figure 1. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized NDR Throughput for Phy-to-Phy SRv6.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[33].

PDR Throughput

VPP PDR 78B packet throughput speedup ratio is presented in the graphs below for 10ge2p1x520 network interface card. PDR measured for 0.5% packet loss ratio.

[33] https://git.fd.io/csit/tree/tests/vpp/perf/srv6?h=rls1804

NIC 10ge2p1x520

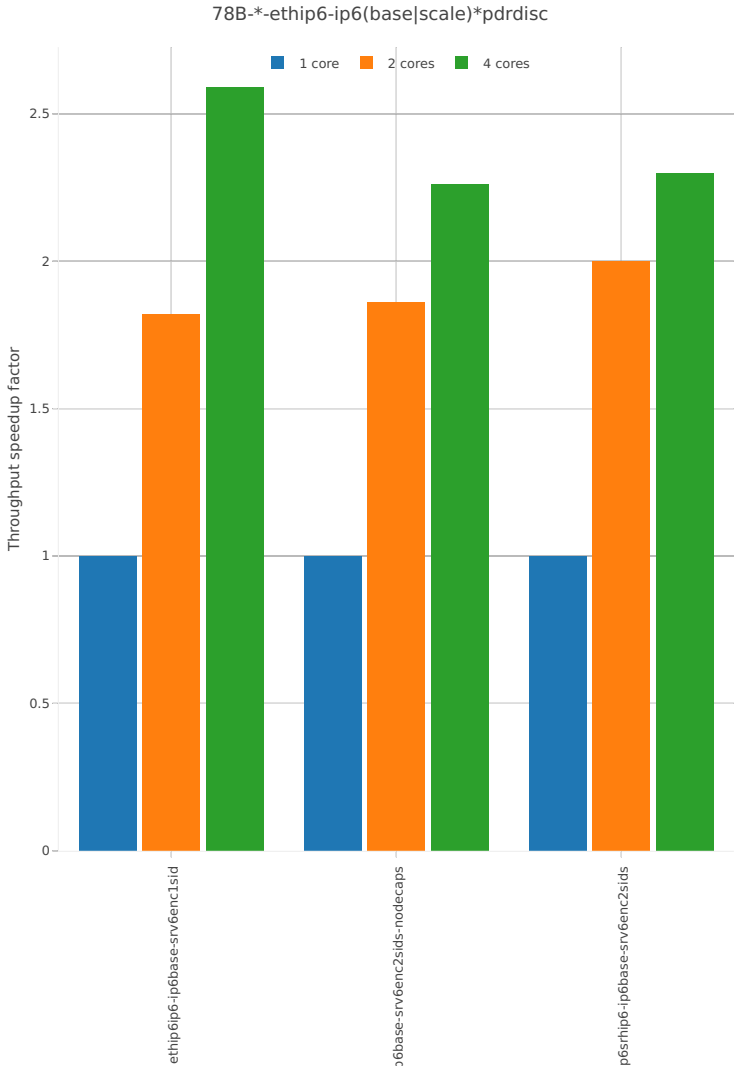


Figure 3. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized PDR Throughput for Phy-to-Phy SRv6.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[34].

2.4.5 IPv4 Overlay Tunnels

Following sections include Throughput Speedup Analysis for VPP multi- core multi-thread configurations with no Hyper-Threading, specifically for tested 2t2c (2threads, 2cores) and 4t4c scenarios. 1t1c throughput results are used as a reference for reported speedup ratio. Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

[34] https://git.fd.io/csit/tree/tests/vpp/perf/srv6?h=rls1804

NDR Throughput

VPP NDR 64B packet throughput speedup ratio is presented in the graphs below for 10ge2p1x520 network interface card.

NIC 10ge2p1x520

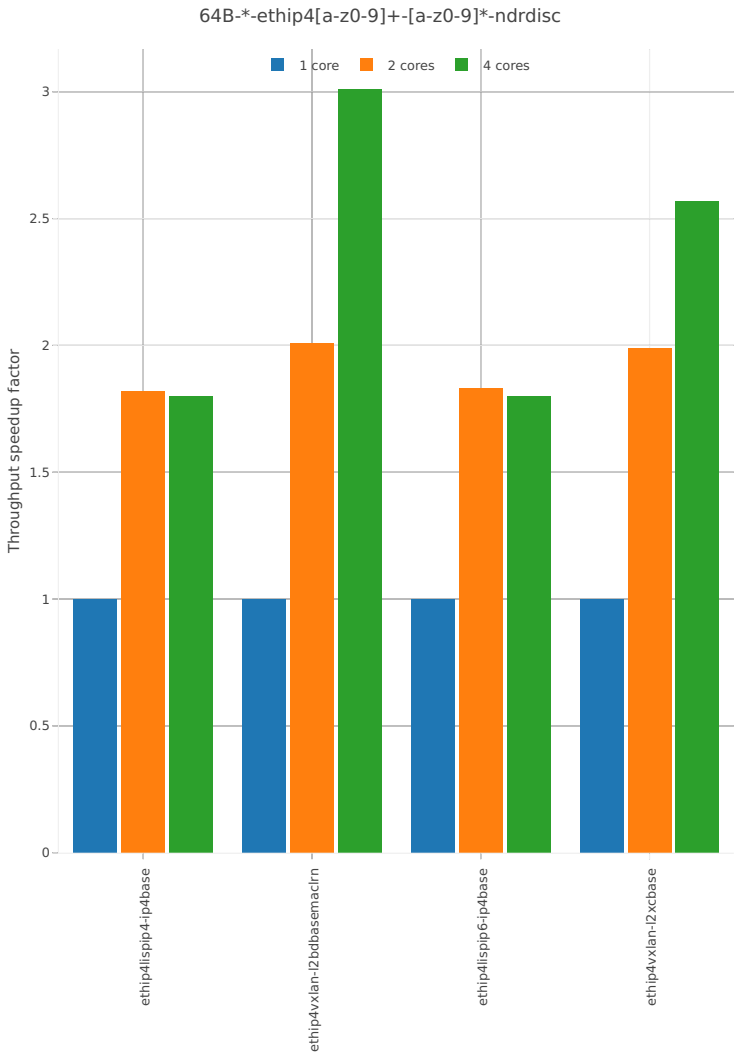


Figure 1. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized NDR Throughput for Phy-to-Phy IPv4 Overlay Tunnels.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[35].

PDR Throughput

VPP PDR 64B packet throughput speedup ratio is presented in the graphs below for 10ge2p1x520 network interface card.

[35] https://git.fd.io/csit/tree/tests/vpp/perf/ip4_tunnels?h=rls1804

NIC 10ge2p1x520

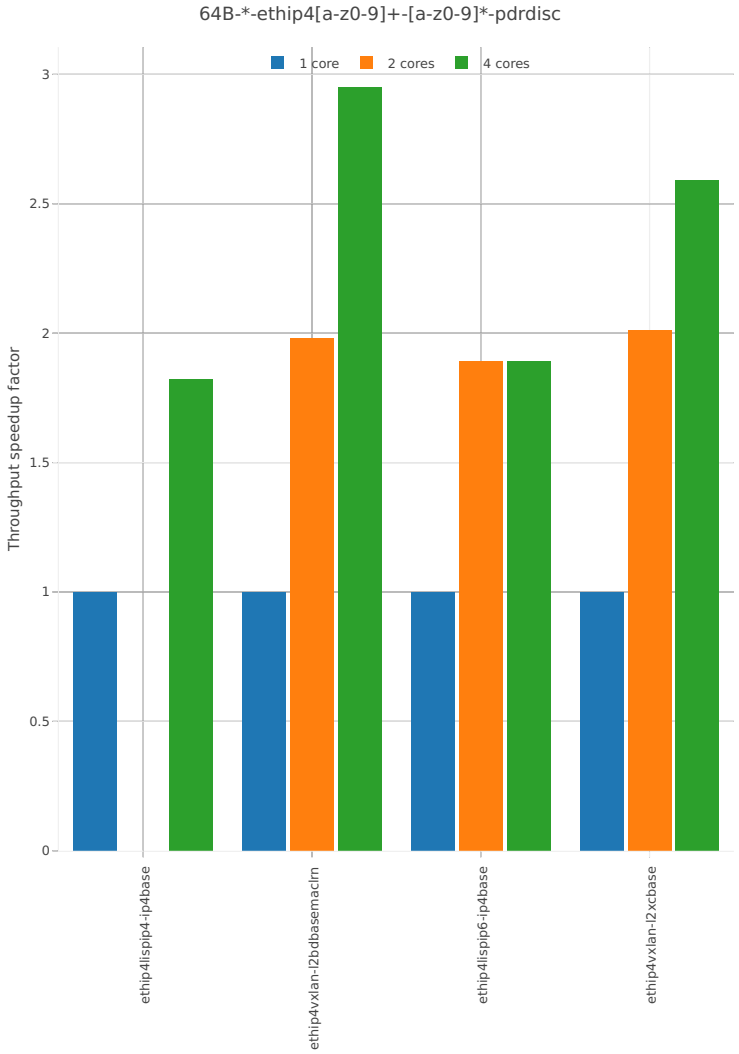


Figure 2. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized PDR Throughput for Phy-to-Phy IPv4 Overlay Tunnels.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[36].

2.4.6 IPv6 Overlay Tunnels

Following sections include Throughput Speedup Analysis for VPP multi- core multi-thread configurations with no Hyper-Threading, specifically for tested 2t2c (2threads, 2cores) and 4t4c scenarios. 1t1c throughput results are used as a reference for reported speedup ratio. Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

[36] https://git.fd.io/csit/tree/tests/vpp/perf/ip4_tunnels?h=rls1804

NDR Throughput

VPP NDR 64B packet throughput speedup ratio is presented in the graphs below for 10ge2p1x520 network interface card.

NIC 10ge2p1x520

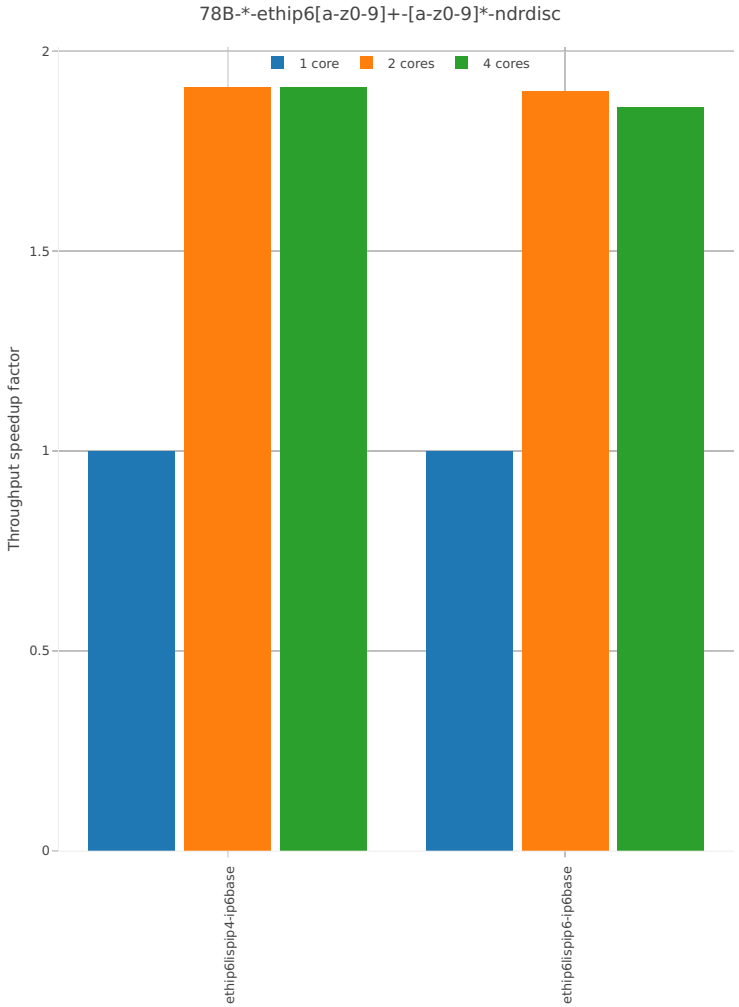


Figure 1. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized NDR Throughput for Phy-to-Phy IPv6 Overlay Tunnels.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[37].

PDR Throughput

VPP PDR 64B packet throughput speedup ratio is presented in the graphs below for 10ge2p1x520 network interface card.

[37] https://git.fd.io/csit/tree/tests/vpp/perf/ip6_tunnels?h=rls1804

NIC 10ge2p1x520

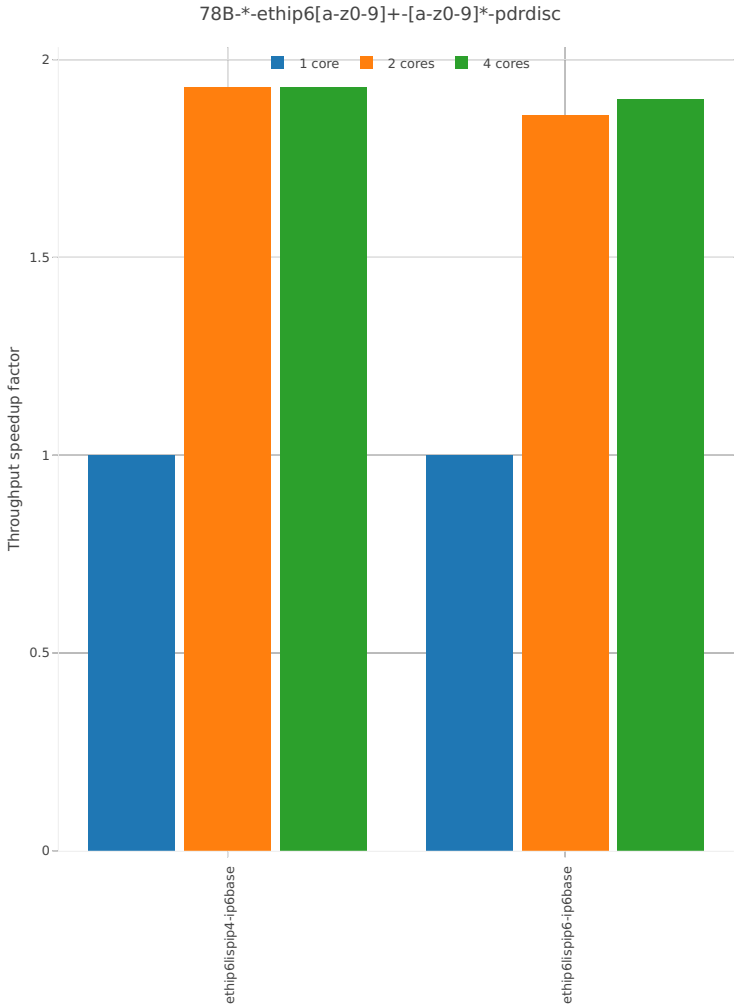


Figure 2. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized PDR Throughput for Phy-to-Phy IPv6 Overlay Tunnels.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[38].

2.4.7 VM vhost Connections

Following sections include Throughput Speedup Analysis for VPP multi- core multi-thread configurations with no Hyper-Threading, specifically for tested 2t2c (2threads, 2cores) and 4t4c scenarios. 1t1c throughput results are used as a reference for reported speedup ratio. Input data used for the graphs comes from Phy-to-Phy 64B performance tests with VM vhost-user, including NDR throughput (zero packet loss) and PDR throughput (<0.5% packet loss).

[38] https://git.fd.io/csit/tree/tests/vpp/perf/ip6_tunnels?h=rls1804

NDR Throughput

VPP NDR 64B packet throughput speedup ratio is presented in the graphs below for 10ge2p1x520, 10ge2p1x710 and 40ge2p1xl710 network interface cards.

NIC 10ge2p1x520

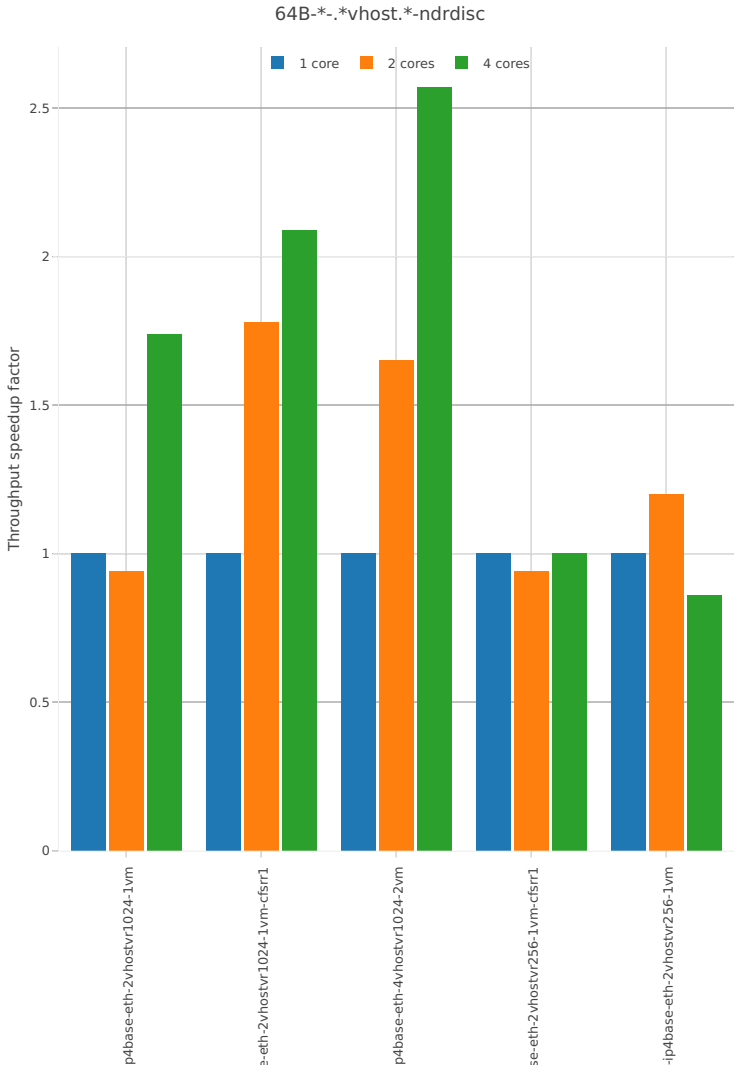


Figure 1a. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized NDR Throughput for Phy-to-Phy VM vhost-user selected TCs.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[39].

[39] https://git.fd.io/csit/tree/tests/vpp/perf/vm_vhost?h=rls1804

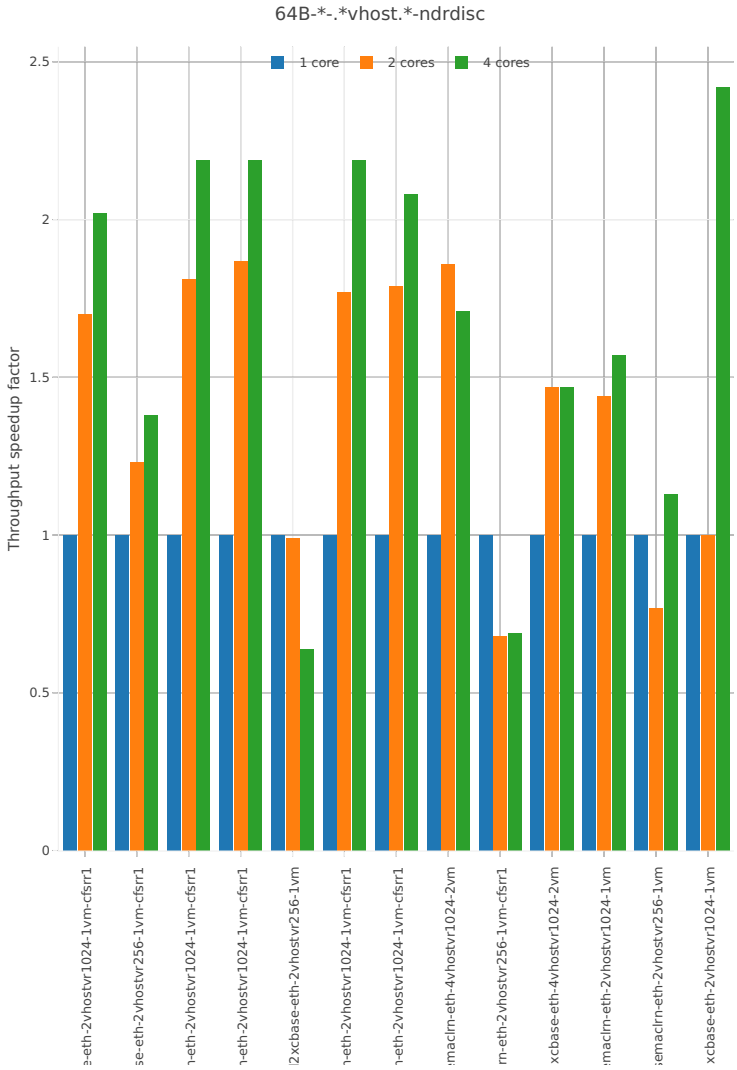


Figure 1b. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized NDR Throughput for Phy-to-Phy VM vhost-user selected TCs.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[40].

[40] https://git.fd.io/csit/tree/tests/vpp/perf/vm_vhost?h=rls1804

NIC 10ge2p1x710

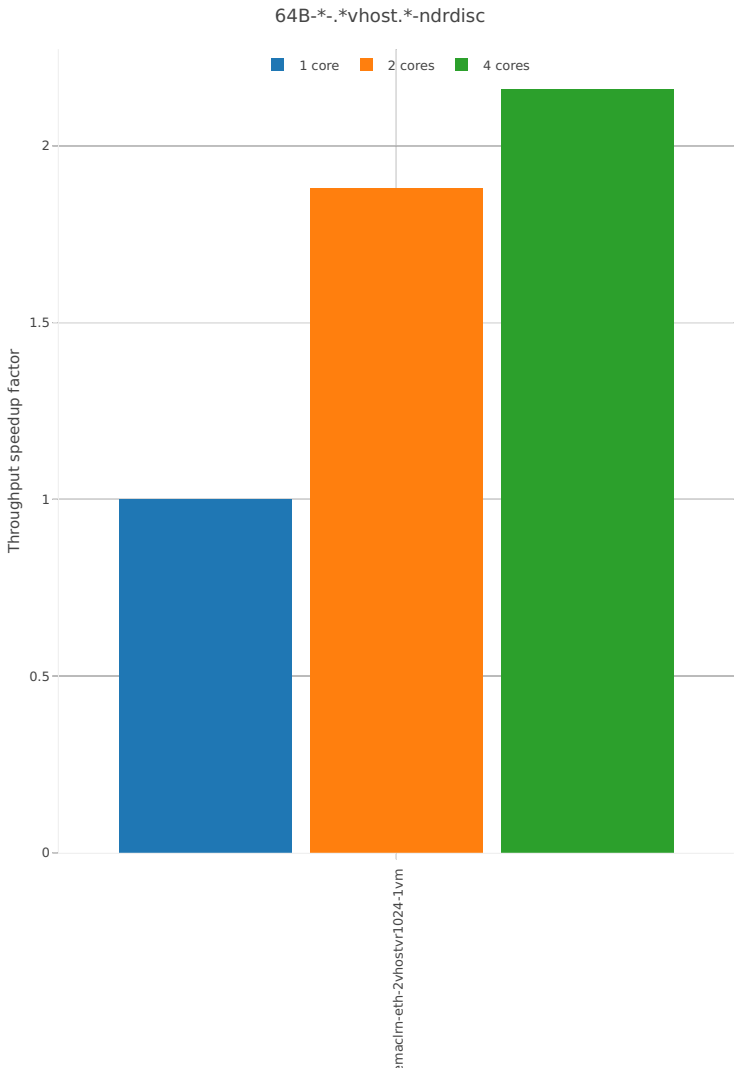


Figure 2. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized NDR Throughput for Phy-to-Phy VM vhost-user selected TCs.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[41].

[41] https://git.fd.io/csit/tree/tests/vpp/perf/vm_vhost?h=rls1804

NIC 40ge2p1xl710

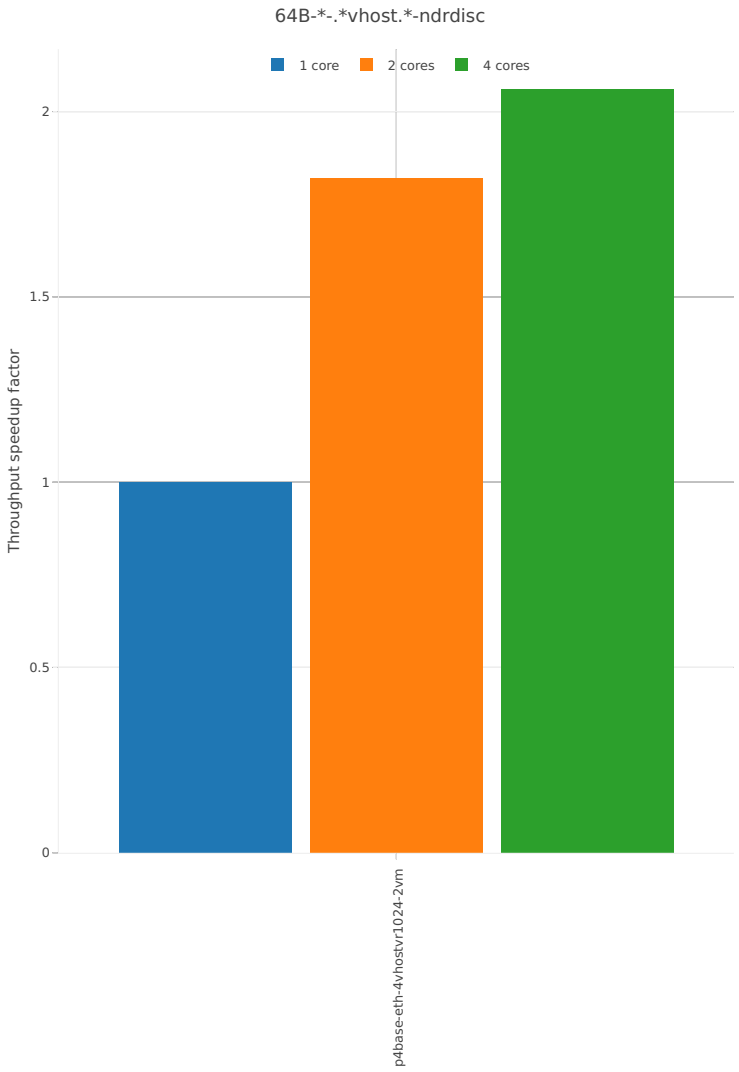


Figure 3a. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized NDR Throughput for Phy-to-Phy VM vhost-user selected TCs.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[42].

[42] https://git.fd.io/csit/tree/tests/vpp/perf/vm_vhost?h=rls1804

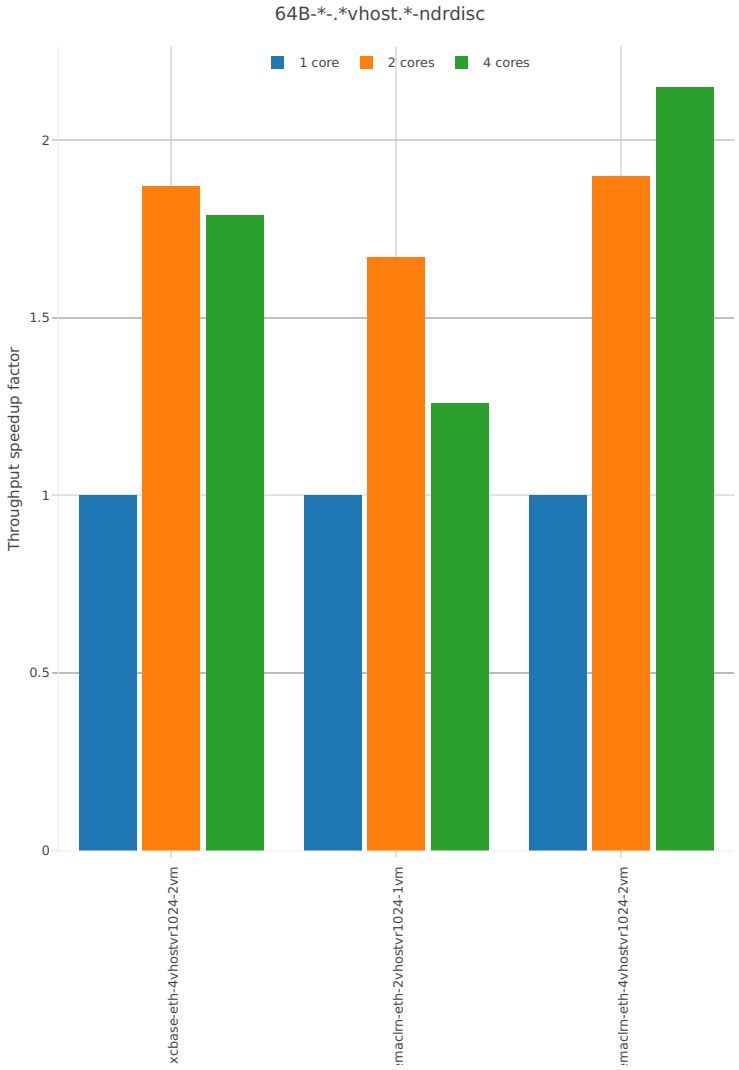


Figure 3b. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized NDR Throughput for Phy-to-Phy VM vhost-user selected TCs.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[43].

PDR Throughput

VPP PDR 64B packet throughput speedup ratio is presented in the graphs below for 10ge2p1x520, 10ge2p1x710 and 40ge2p1xl710 network interface cards.

[43] https://git.fd.io/csit/tree/tests/vpp/perf/vm_vhost?h=rls1804

NIC 10ge2p1x520

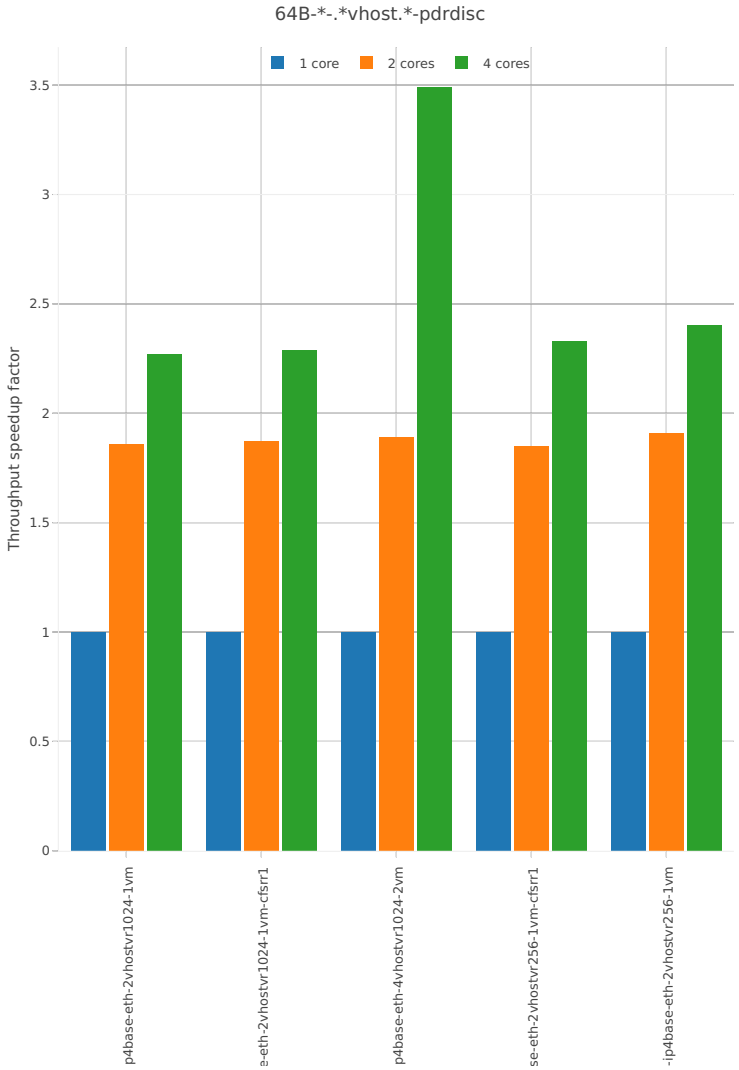


Figure 4a. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized PDR Throughput for Phy-to-Phy VM vhost-user selected TCs.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[44].

[44] https://git.fd.io/csit/tree/tests/vpp/perf/vm_vhost?h=rls1804

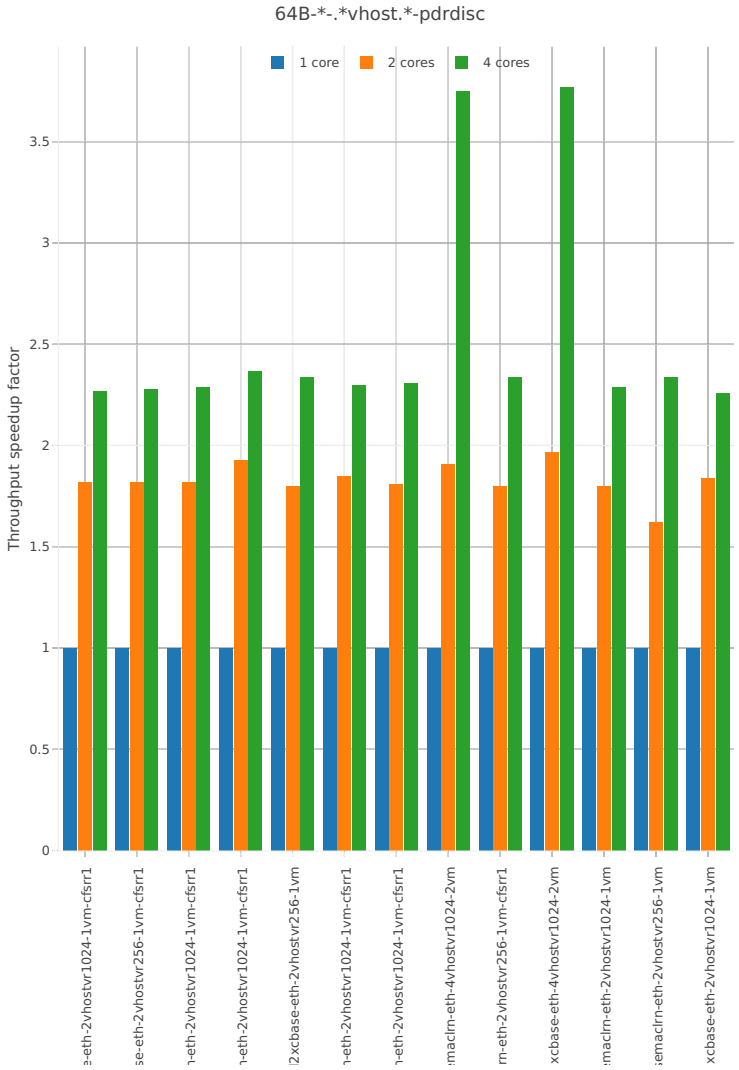


Figure 4b. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized PDR Throughput for Phy-to-Phy VM vhost-user selected TCs.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[45].

[45] https://git.fd.io/csit/tree/tests/vpp/perf/vm_vhost?h=rls1804

NIC 10ge2p1x710

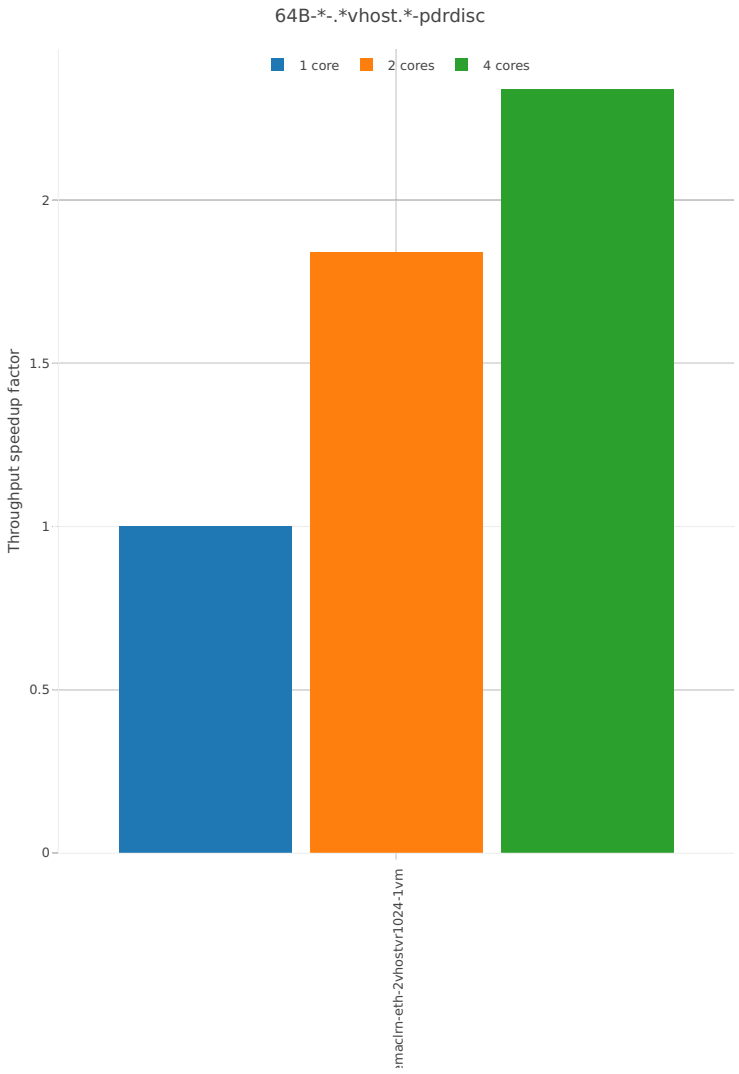


Figure 5. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized PDR Throughput for Phy-to-Phy VM vhost-user selected TCs.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[46].

[46] https://git.fd.io/csit/tree/tests/vpp/perf/vm_vhost?h=rls1804

NIC 40ge2p1xl710

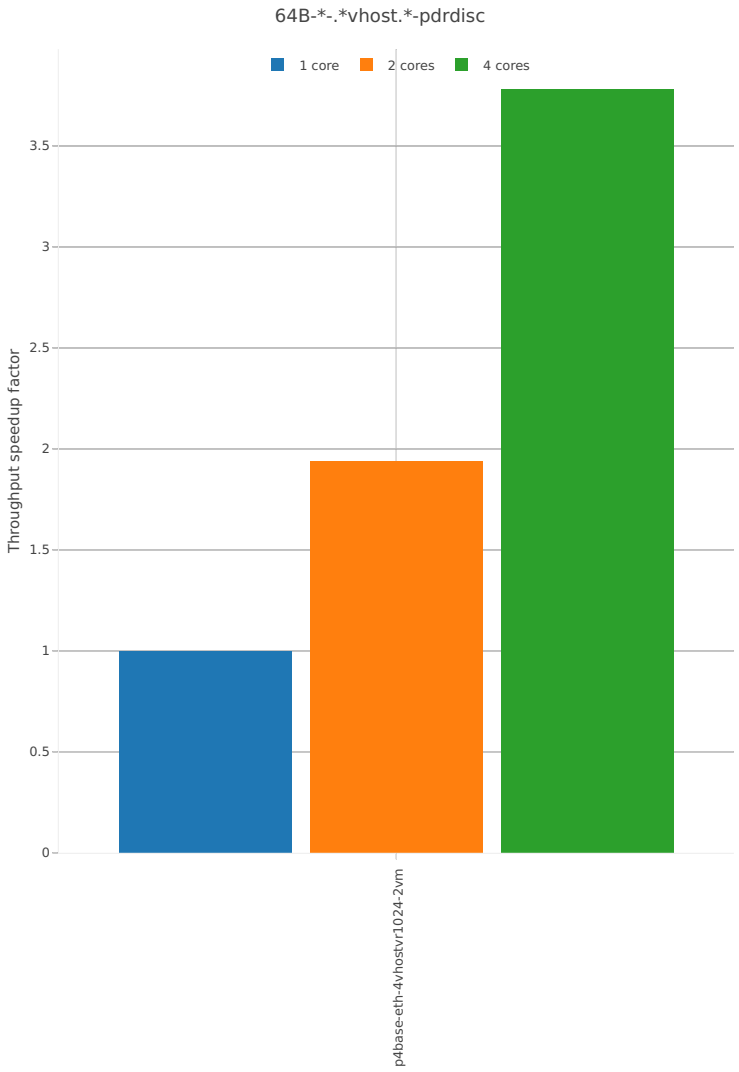


Figure 6a. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized PDR Throughput for Phy-to-Phy VM vhost-user selected TCs.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[47].

[47] https://git.fd.io/csit/tree/tests/vpp/perf/vm_vhost?h=rls1804

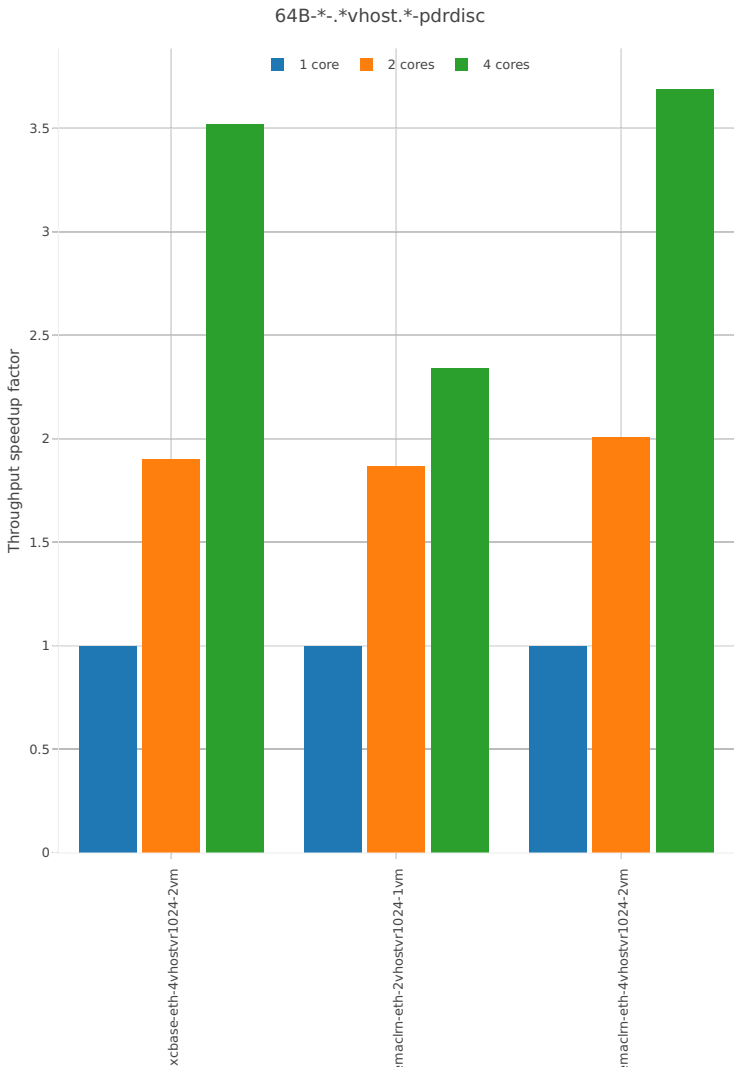


Figure 6b. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized PDR Throughput for Phy-to-Phy VM vhost-user selected TCs.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[48].

2.4.8 Container memif Connections

Following sections include Throughput Speedup Analysis for VPP multi- core multi-thread configurations with no Hyper-Threading, specifically for tested 2t2c (2threads, 2cores) and 4t4c scenarios. 1t1c throughput results are used as a reference for reported speedup ratio. Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

NDR Throughput

VPP NDR 64B packet throughput speedup ratio is presented in the graphs below for 10ge2p1x520 network interface card.

[48] https://git.fd.io/csit/tree/tests/vpp/perf/vm_vhost?h=rls1804

NIC 10ge2p1x520

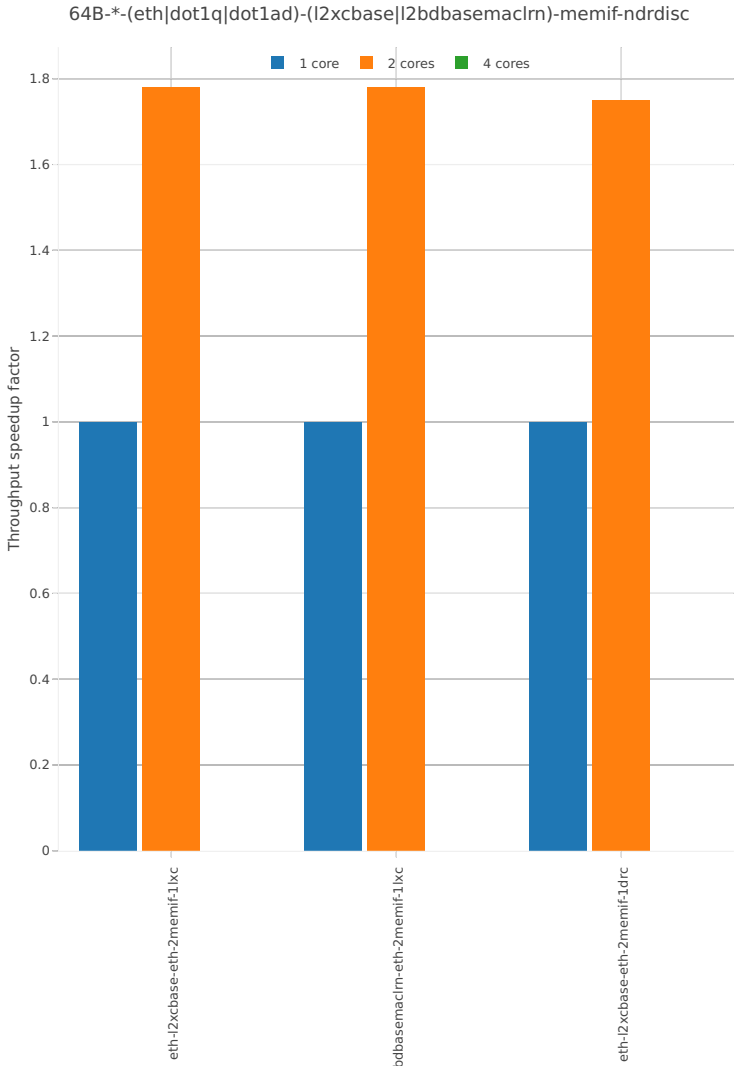


Figure 1. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized NDR Throughput for Phy-to-Phy L2 Ethernet Switching (base).

CSIT source code for the test cases used for above plots can be found in CSIT git repository[49].

PDR Throughput

VPP PDR 64B packet throughput speedup ratio is presented in the graphs below for 10ge2p1x520 network interface card.

[49] https://git.fd.io/csit/tree/tests/vpp/perf/container_memif?h=rls1804

NIC 10ge2p1x520

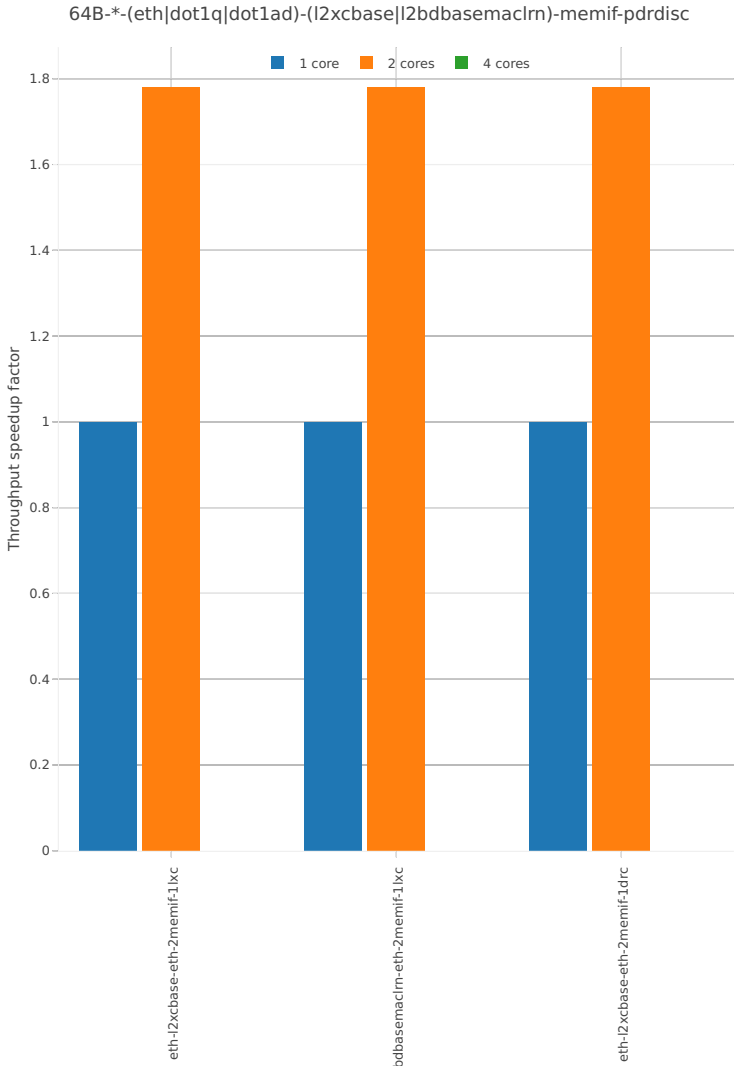


Figure 2. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized PDR Throughput for Phy-to-Phy L2 Ethernet Switching (base).

CSIT source code for the test cases used for above plots can be found in CSIT git repository[50].

2.4.9 Container Orchestrated Topologies

Following sections include Throughput Speedup Analysis for VPP multi- core multi-thread configurations with no Hyper-Threading, specifically for tested 2t2c (2threads, 2cores) and 4t4c scenarios. 1t1c throughput results are used as a reference for reported speedup ratio. Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

[50] https://git.fd.io/csit/tree/tests/vpp/perf/container_memif?h=rls1804

NDR Throughput

VPP NDR 64B packet throughput speedup ratio is presented in the graphs below for 10ge2p1x520 and 10ge2p1x710 network interface cards.

NIC 10ge2p1x520

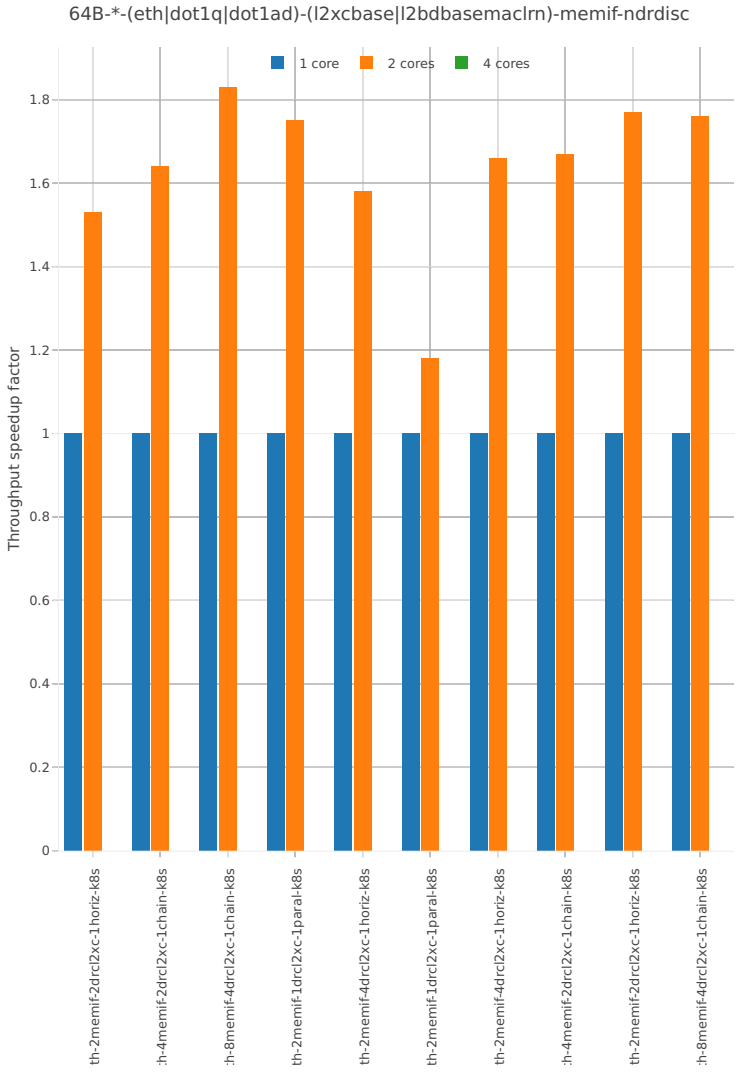


Figure 1. VPP 1thread 1core - NDR Throughput for Phy-to-Phy L2 Ethernet Switching (base).

CSIT source code for the test cases used for above plots can be found in CSIT git repository[51].

NIC 10ge2p1x710

Figure 2. VPP 1thread 1core - NDR Throughput for Phy-to-Phy L2 Ethernet Switching (base).

[51] https://git.fd.io/csit/tree/tests/kubernetes/perf/container_memif?h=rls1804

CSIT source code for the test cases used for above plots can be found in CSIT git repository[52].

PDR Throughput

VPP PDR 64B packet throughput speedup ratio is presented in the graphs below for 10ge2p1x520 and 10ge2p1x710 network interface cards.

NIC 10ge2p1x520

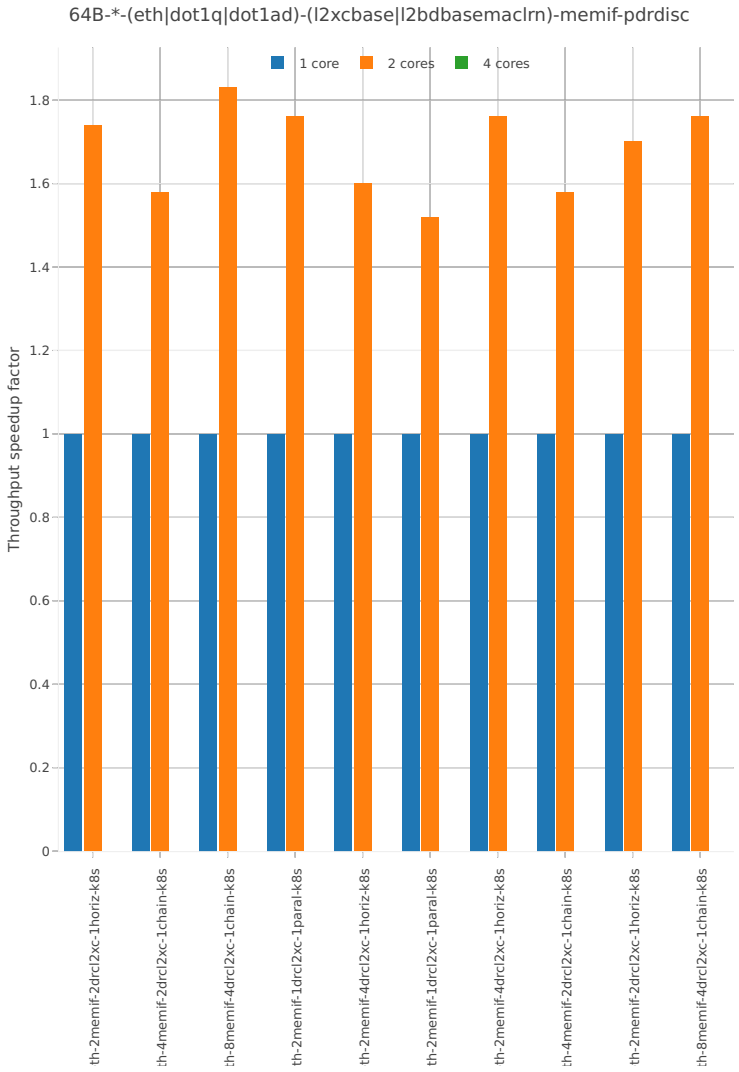


Figure 3. VPP 1thread 1core - NDR Throughput for Phy-to-Phy L2 Ethernet Switching (base).

CSIT source code for the test cases used for above plots can be found in CSIT git repository[53].

NIC 10ge2p1x710

[52] https://git.fd.io/csit/tree/tests/kubernetes/perf/container_memif?h=rls1804
[53] https://git.fd.io/csit/tree/tests/kubernetes/perf/container_memif?h=rls1804

Figure 4. VPP 1thread 1core - NDR Throughput for Phy-to-Phy L2 Ethernet Switching (base).

CSIT source code for the test cases used for above plots can be found in CSIT git repository[54].

2.4.10 IPSec Crypto HW: IP4 Routed-Forwarding

Following sections include Throughput Speedup Analysis for VPP multi- core multi-thread configurations with no Hyper-Threading, specifically for tested 2t2c (2threads, 2cores) and 4t4c scenarios. 1t1c throughput results are used as a reference for reported speedup ratio. VPP IPSec encryption is accelerated using DPDK cryptodev library driving Intel Quick Assist (QAT) crypto PCIe hardware cards. Performance is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

NDR Throughput

VPP NDR 64B packet throughput speedup ratio is presented in the graphs below for 40ge2p1xl710 network interface card.

[54] https://git.fd.io/csit/tree/tests/kubernetes/perf/container_memif?h=rls1804

NIC 40ge2p1xl710

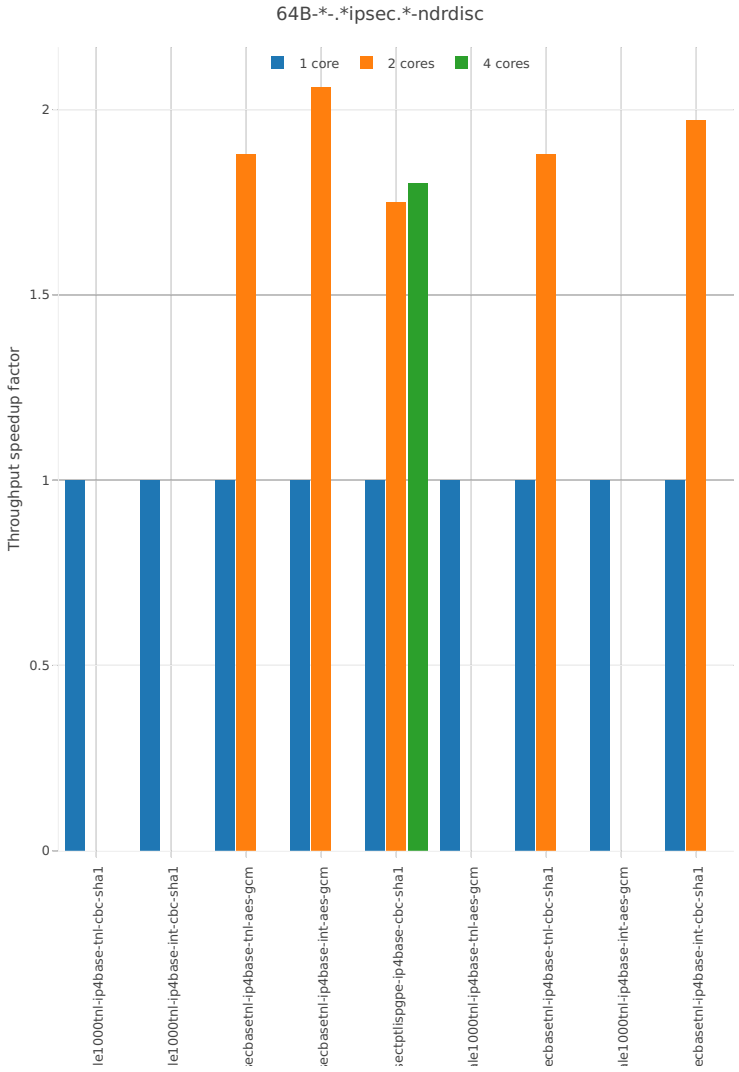


Figure 1. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized NDR Throughput for Phy-to-Phy IPSEC HW.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[55].

PDR Throughput

VPP PDR 64B packet throughput speedup ratio is presented in the graphs below for 40ge2p1xl710 network interface card.

NIC 40ge2p1xl710

VPP PDR 64B packet throughput in 1t1c setup (1thread, 1core) is presented in the graph below. PDR measured for 0.5% packet loss ratio.

[55] https://git.fd.io/csit/tree/tests/vpp/perf/crypto?h=rls1804

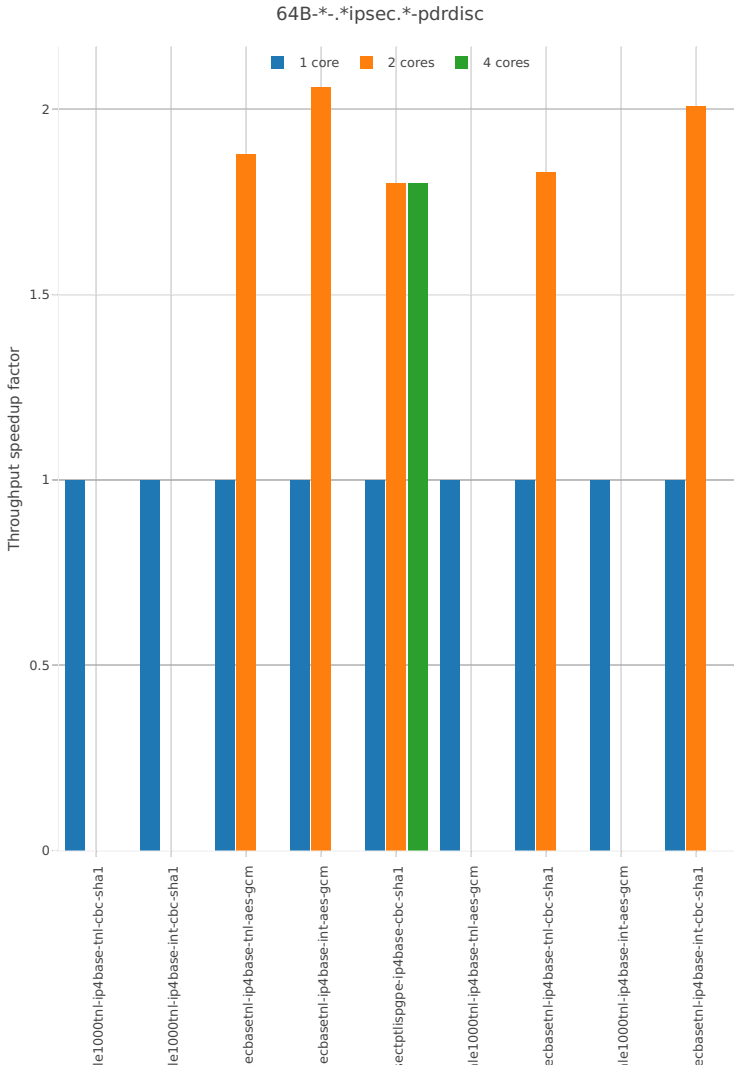


Figure 2. Throughput Speedup Analysis - Multi-Core Speedup Ratio - Normalized PDR Throughput for Phy-to-Phy IPSEC HW.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[56].

2.5 Packet Latency Graphs

Plotted results are generated from a single execution of CSIT NDR discovery test. Box plots are used to show the Minimum, Median and Maximum packet latency per test.

Title of each graph is a regex (regular expression) matching all throughput test cases plotted on this graph, *X-axis labels* are indices of individual test suites executed by FD.io test executor vpp performance jobs[57] that created result output file used as data source for the graph, *Y-axis labels* are measured packet Latency [uSec] values, and the *Graph legend* lists the plotted test suites and their indices. Latency is reported for concurrent symmetric bi-directional flows, separately for each direction: i) West-to-East: TGint1-to-SUT1-to-SUT2-to-TGint2, and ii) East-to-West: TGint2-to-SUT2-to-SUT1-to-TGint1.

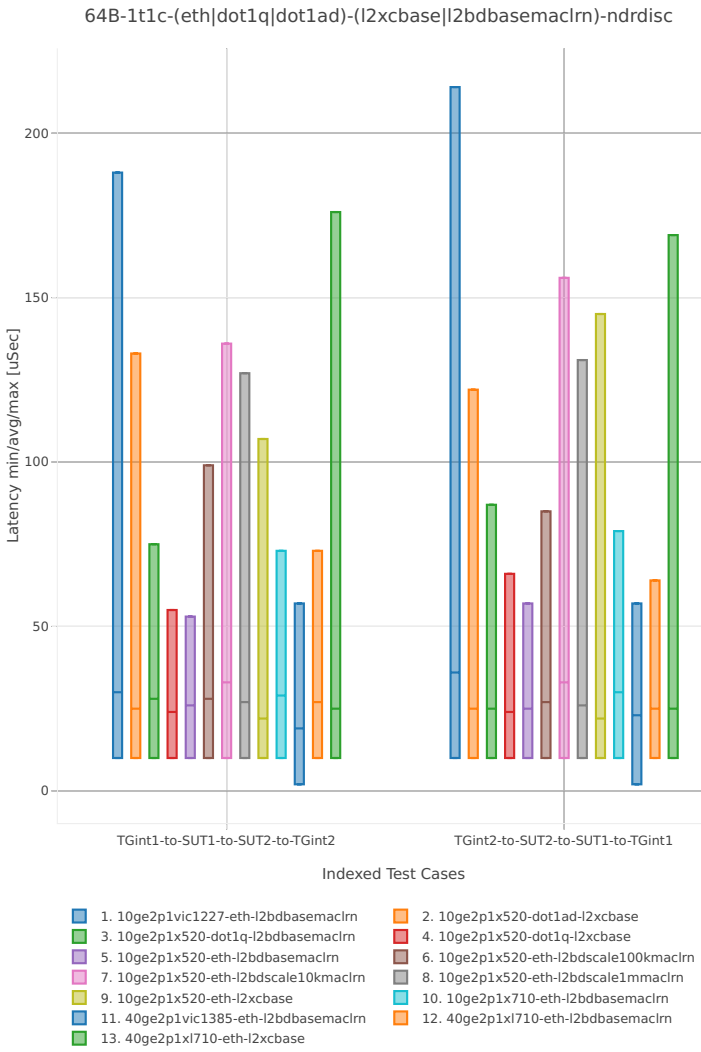
[56] https://git.fd.io/csit/tree/tests/vpp/perf/crypto?h=rls1804
[57] https://jenkins.fd.io/view/csit/job/csit-vpp-perf-1801-all

Note: Test results have been generated by FD.io test executor vpp performance jobs[58] with Robot Framework result files csit-vpp-perf-1801-*.zip archived here.

2.5.1 L2 Ethernet Switching

This section includes summary graphs of VPP Phy-to-Phy packet latency with L2 Ethernet switching measured at 50% of discovered NDR throughput rate. Latency is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

VPP packet latency in 1t1c setup (1thread, 1core) is presented in the graph below.



CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ grep -E "64B-1t1c-(eth|dot1q|dot1ad)-(l2xcbase|l2bdbasemaclrn|l2bdscale.*|l2dbscale.*)-(eth.
↪*)*ndrdisc" tests/vpp/perf/l2/* && grep -E "64B-1t1c-(eth|dot1q|dot1ad)-
↪(l2xcbase|l2bdbasemaclrn|l2bdscale.*)-(eth.*)*ndrdisc" tests/vpp/perf/container_memif/*
```

[58] https://jenkins.fd.io/view/csit/job/csit-vpp-perf-1801-all

Figure 1a. VPP 1thread 1core - packet latency for Phy-to-Phy L2 Ethernet Switching (base).

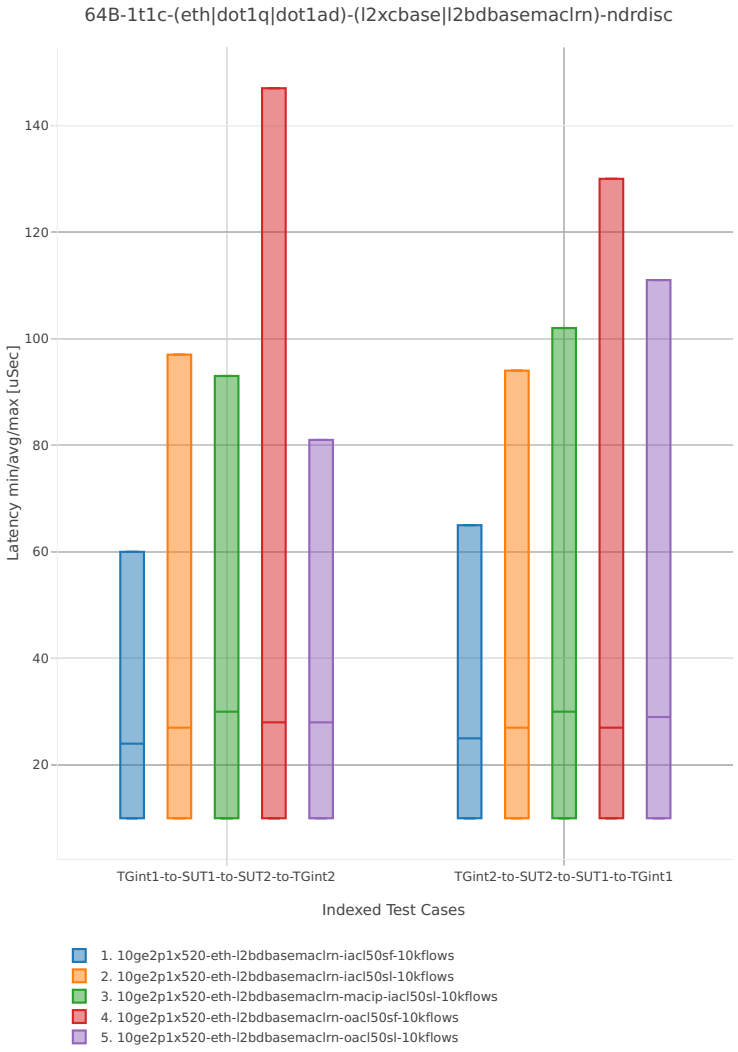
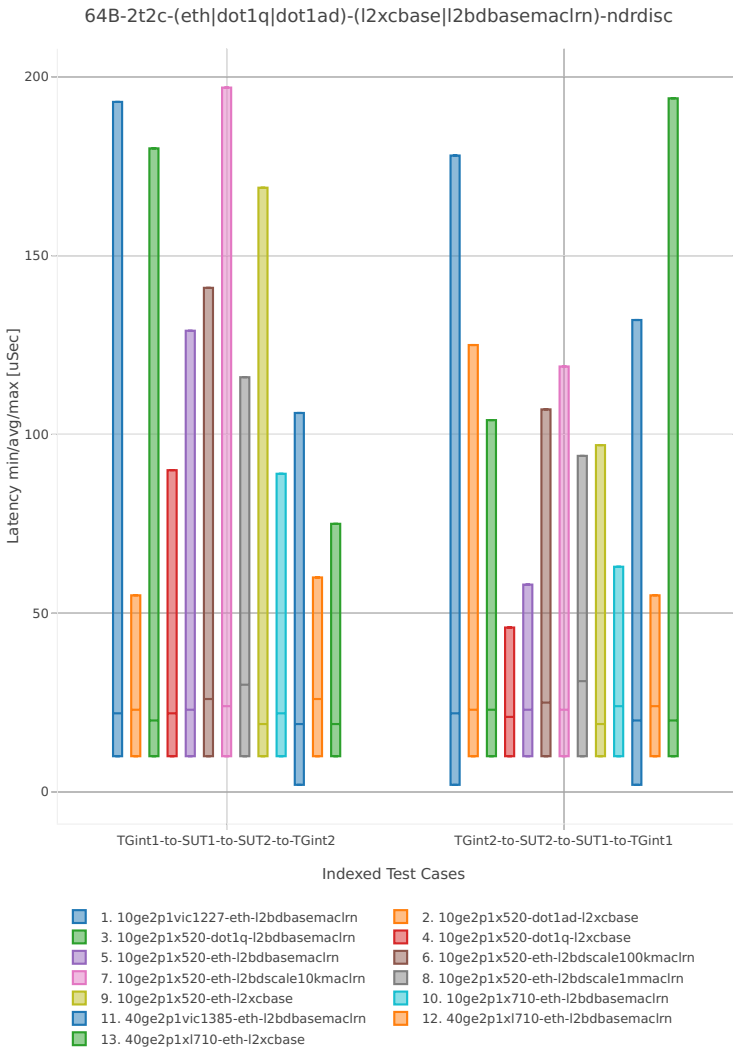


Figure 1b. VPP 1thread 1core - packet latency for Phy-to-Phy L2 Ethernet Switching (feature).

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/l2
$ grep -E "64B-1t1c-(eth|dot1q|dot1ad)-(l2xcbase|l2bdbasemaclrn).*-(iacl50(-state(ful|less)|sl)-
↪(flows10k.*|10kflows.*)|oacl50-state(ful|less)-flows10k.*)-ndrdisc" *
```

VPP packet latency in 2t2c setup (2thread, 2core) is presented in the graph below.



CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ grep -E "64B-2t2c-(eth|dot1q|dot1ad)-(l2xcbase|l2bdbasemaclrn|l2bdscale.*|l2dbscale.*)-(eth.
↪*)*ndrdisc" tests/vpp/perf/l2/* && grep -E "64B-2t2c-(eth|dot1q|dot1ad)-
↪(l2xcbase|l2bdbasemaclrn|l2bdscale.*)-(eth.*)*ndrdisc" tests/vpp/perf/container_memif/*
```

Figure 2a. VPP 2threads 2cores - packet latency for Phy-to-Phy L2 Ethernet Switching (base).

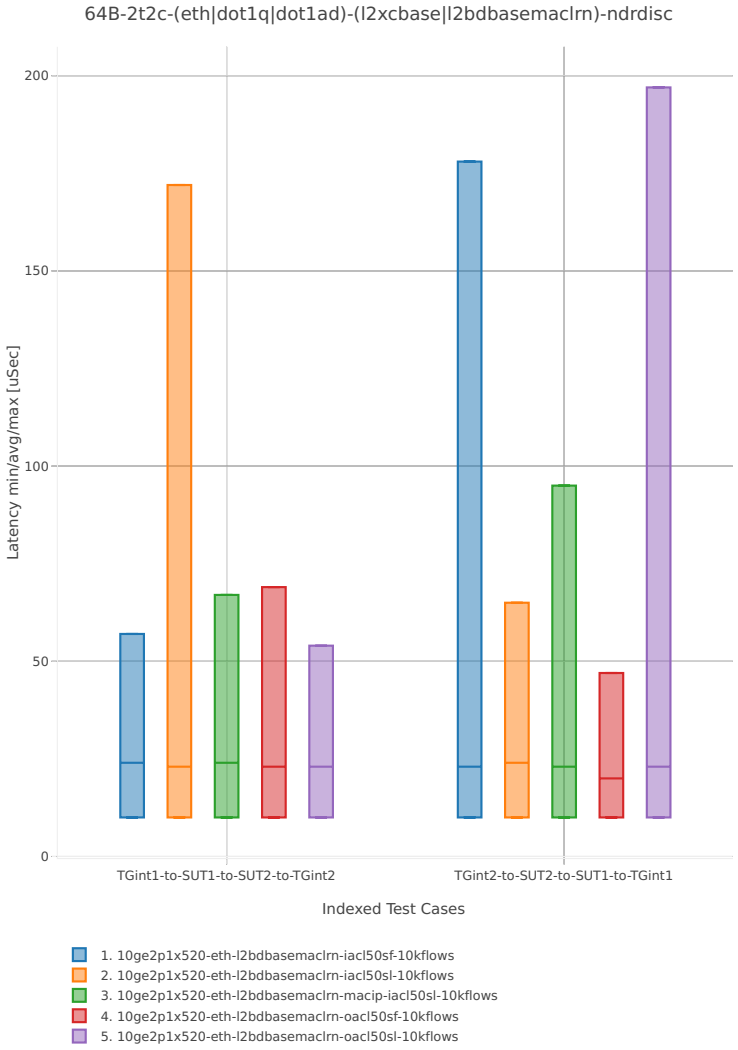


Figure 2b. VPP 2threads 2cores - packet latency for Phy-to-Phy L2 Ethernet Switching (feature).

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/l2
$ grep -E "64B-2t2c-(eth|dot1q|dot1ad)-(l2xcbase|l2bdbasemaclrn).*-(iacl50(-state(ful|less)|sl)-
↪(flows10k.*|10kflows.*)|oacl50-state(ful|less)-flows10k.*)-ndrdisc" *
```

2.5.2 IPv4 Routed-Forwarding

This section includes summary graphs of VPP Phy-to-Phy packet latency with IPv4 Routed-Forwarding measured at 50% of discovered NDR throughput rate. Latency is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

VPP packet latency in 1t1c setup (1thread, 1core) is presented in the graph below.

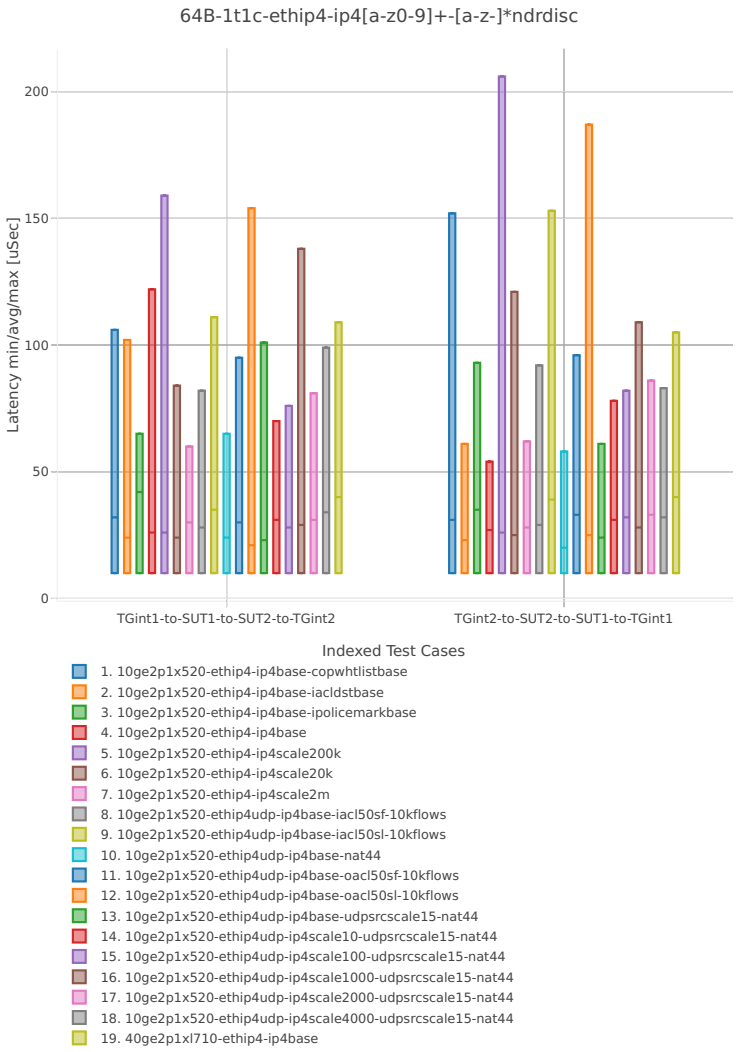


Figure 1. VPP 1thread 1core - packet latency for Phy-to-Phy IPv4 Routed-Forwarding.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/ip4
$ grep -P '64B-1t1c-ethip4(udp)*-ip4(base|scale[a-z0-9]*)(-iacl50-state(ful|less)-flows10k.*|-
↪oacl50-state(ful|less)-flows10k.*|-snat.*|-udp.*|-cop.*|-iacldst.*|-ipolice.*)*-ndrdisc' *
```

VPP packet latency in 2t2c setup (2thread, 2core) is presented in the graph below.

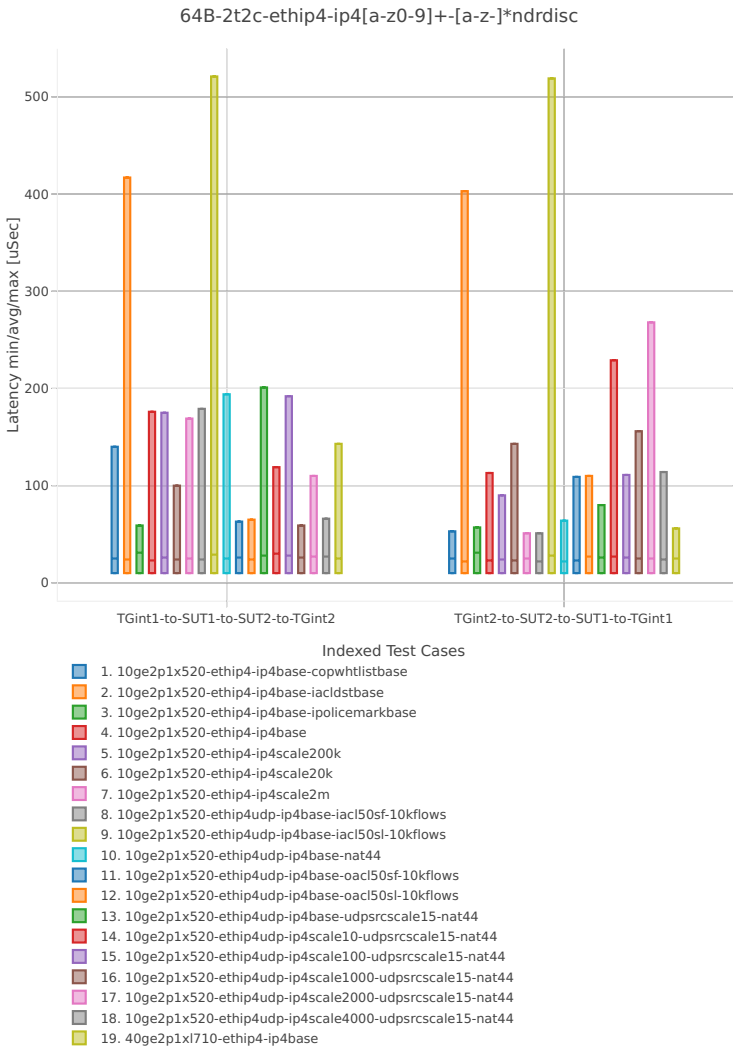


Figure 2. VPP 2threads 2cores - packet latency for Phy-to-Phy IPv4 Routed-Forwarding.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/ip4
$ grep -P '64B-2t2c-ethip4(udp)*-ip4(base|scale[a-z0-9]*)(-iacl50-state(ful|less)-flows10k.*|-
↪oacl50-state(ful|less)-flows10k.*|-snat.*|-udp.*|-cop.*|-iacldst.*|-ipolice.*)*-ndrdisc' *
```

2.5.3 IPv6 Routed-Forwarding

This section includes summary graphs of VPP Phy-to-Phy packet latency with IPv6 Routed-Forwarding measured at 50% of discovered NDR throughput rate. Latency is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

VPP packet latency in 1t1c setup (1thread, 1core) is presented in the graph below.

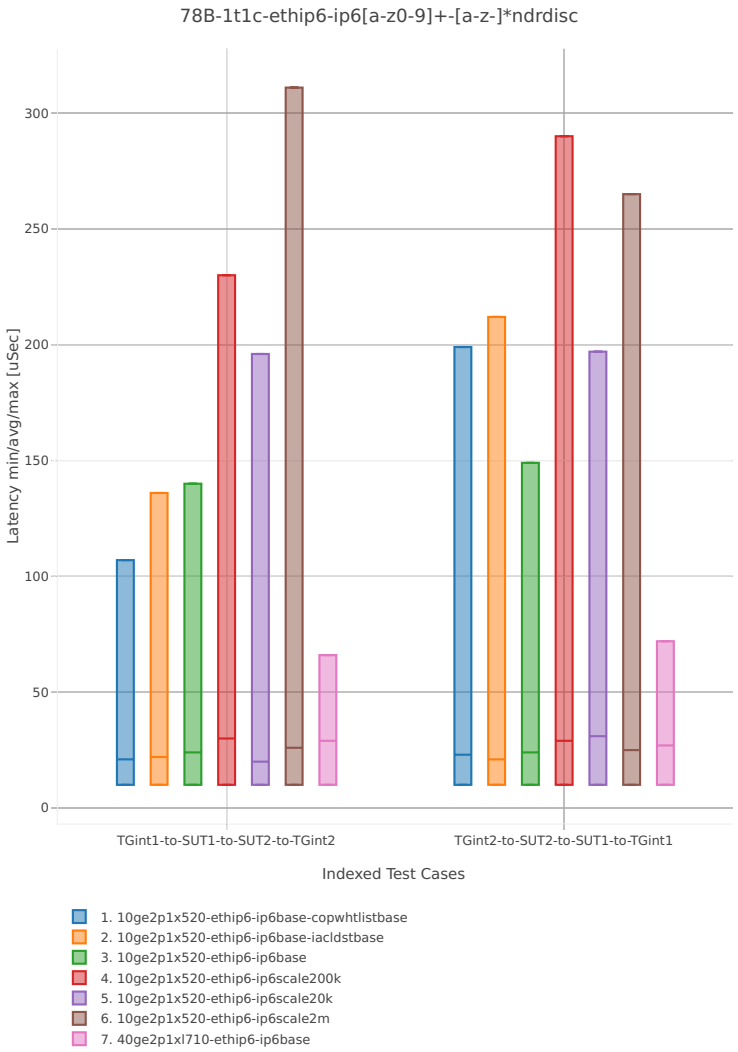


Figure 1. VPP 1thread 1core - packet latency for Phy-to-Phy IPv6 Routed-Forwarding.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/ip6
$ grep -E "78B-1t1c-ethip6-ip6[a-z0-9]+-[a-z-]*ndrdisc" *
```

VPP packet latency in 2t2c setup (2thread, 2core) is presented in the graph below.

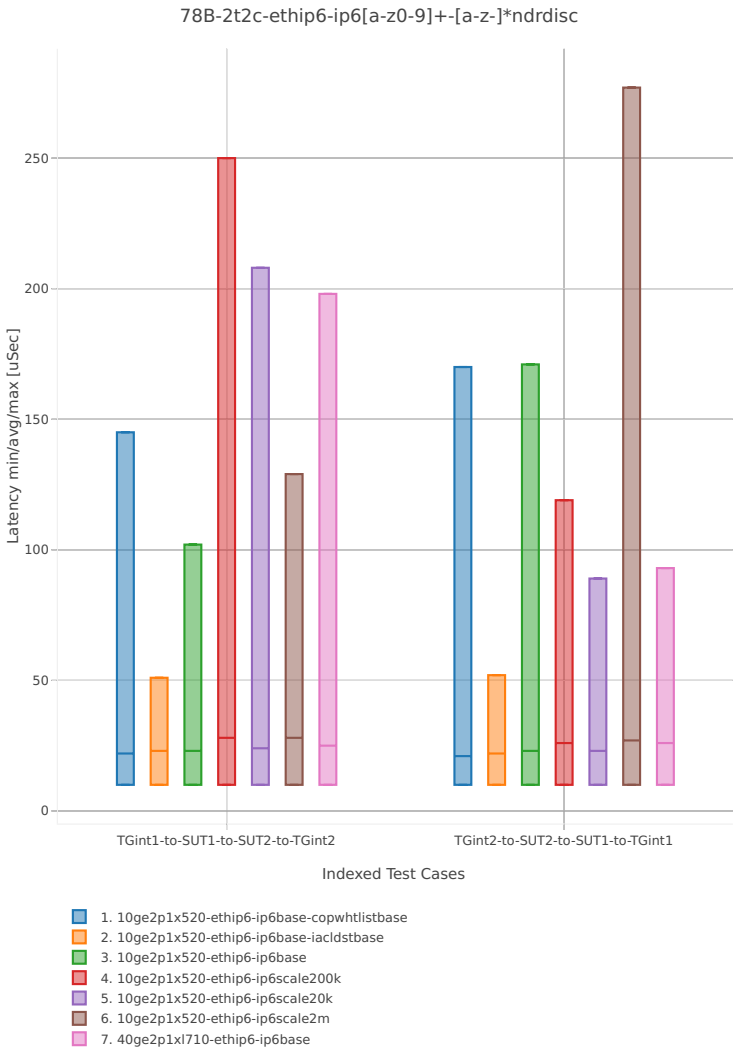


Figure 2. VPP 2threads 2cores - packet latency for Phy-to-Phy IPv6 Routed-Forwarding.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/ip6
$ grep -E "78B-2t2c-ethip6-ip6[a-z0-9]+-[a-z-]*ndrdisc" *
```

2.5.4 SRv6

This section includes summary graphs of VPP Phy-to-Phy packet latency with SRv6 measured at 50% of discovered NDR throughput rate. Latency is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

VPP packet latency in 1t1c setup (1thread, 1core) is presented in the graph below.

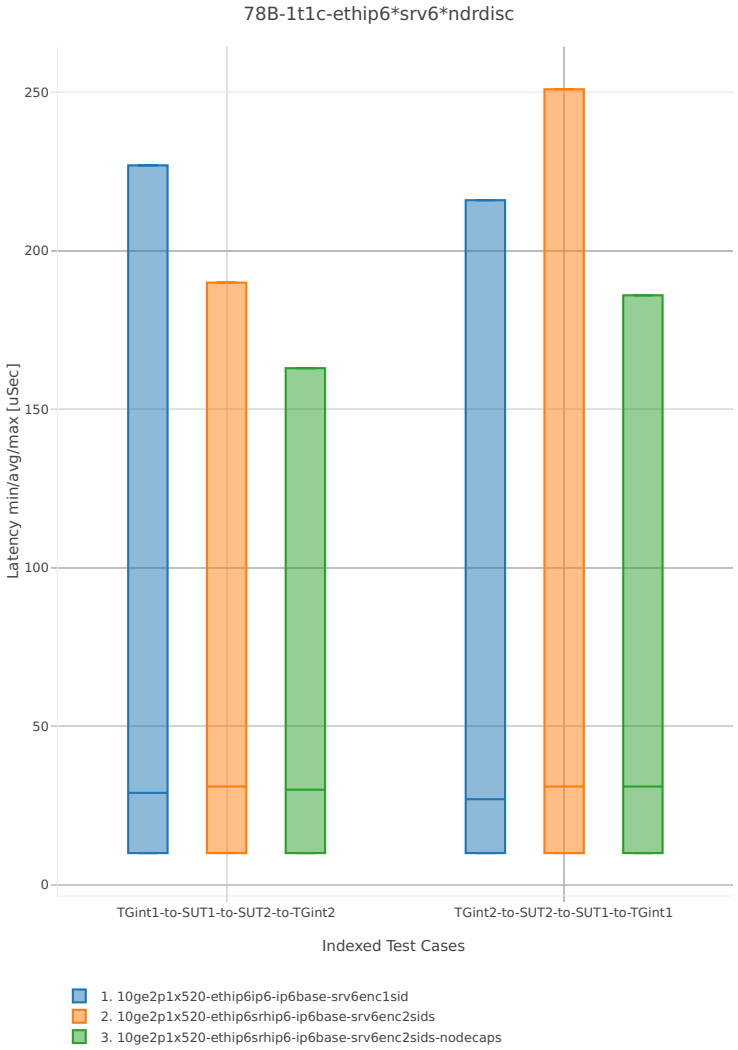


Figure 1. VPP 1thread 1core - packet latency for Phy-to-Phy SRv6.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[59].

VPP packet latency in 2t2c setup (2thread, 2core) is presented in the graph below.

[59] https://git.fd.io/csit/tree/tests/vpp/perf/srv6?h=rls1804

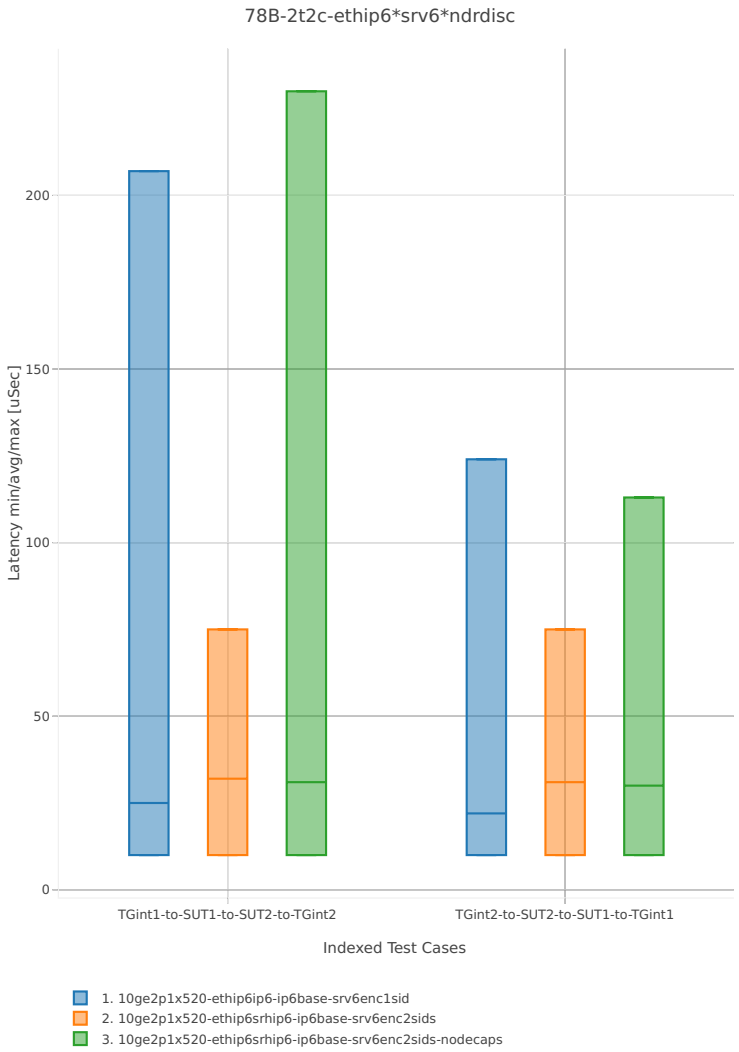


Figure 2. VPP 2threads 2cores - packet latency for Phy-to-Phy SRv6.

CSIT source code for the test cases used for above plots can be found in CSIT git repository[60].

2.5.5 IPv4 Overlay Tunnels

This section includes summary graphs of VPP Phy-to-Phy packet latency with IPv4 Overlay Tunnels measured at 50% of discovered NDR throughput rate. Latency is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

VPP packet latency in 1t1c setup (1thread, 1core) is presented in the graph below.

[60] https://git.fd.io/csit/tree/tests/vpp/perf/srv6?h=rls1804

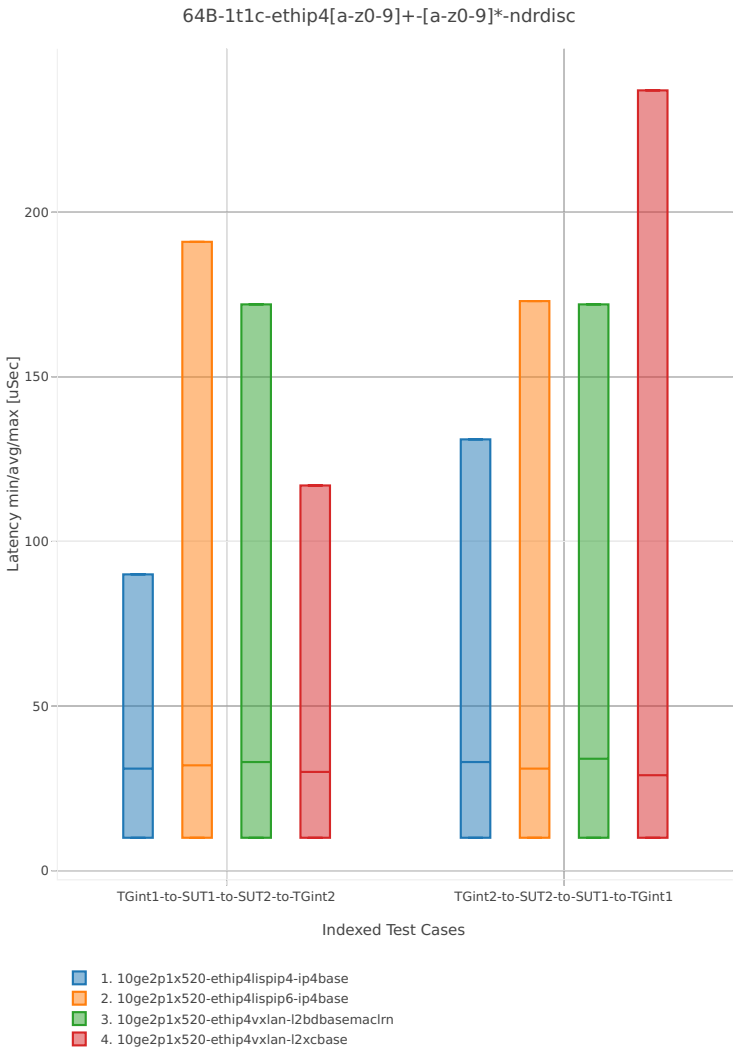


Figure 1. VPP 1thread 1core - packet latency for Phy-to-Phy IPv4 Overlay Tunnels.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/ip4_tunnels
$ grep -E "64B-1t1c-ethip4[a-z0-9]+-[a-z0-9]*-ndrdisc" *
```

VPP packet latency in 2t2c setup (2thread, 2core) is presented in the graph below.

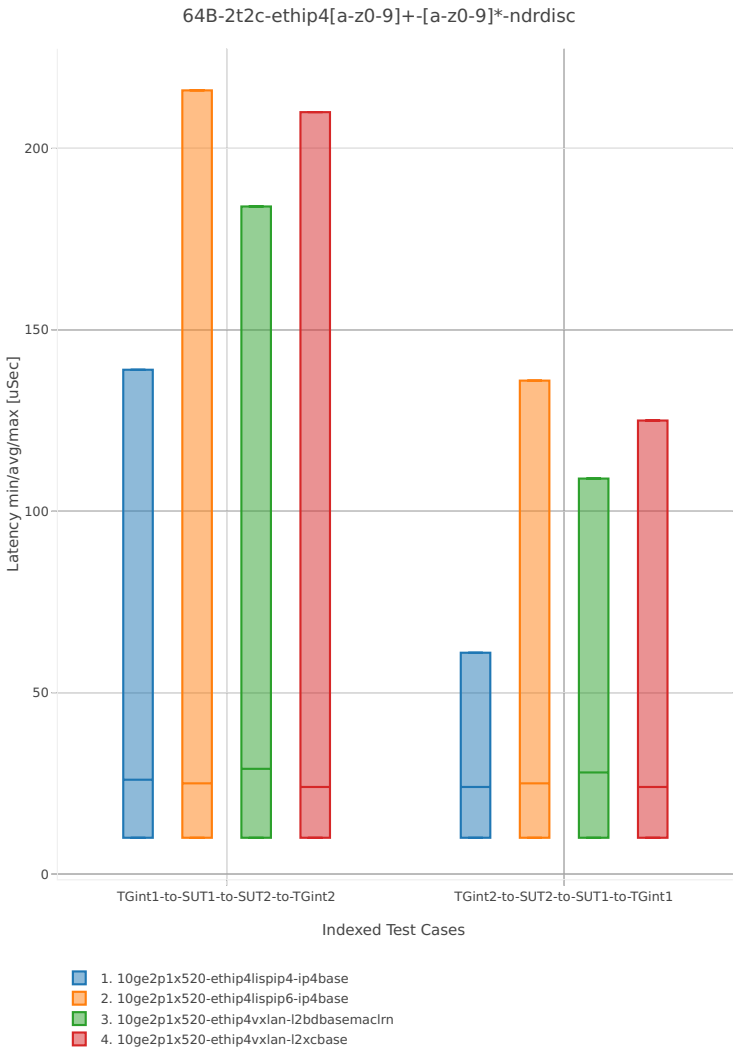


Figure 2. VPP 2threads 2cores - packet latency for Phy-to-Phy IPv4 Overlay Tunnels.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/ip4_tunnels
$ grep -E "64B-2t2c-ethip4[a-z0-9]+-[a-z0-9]*-ndrdisc" *
```

2.5.6 IPv6 Overlay Tunnels

This section includes summary graphs of VPP Phy-to-Phy packet latency with IPv6 Overlay Tunnels measured at 50% of discovered NDR throughput rate. Latency is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

VPP packet latency in 1t1c setup (1thread, 1core) is presented in the graph below.

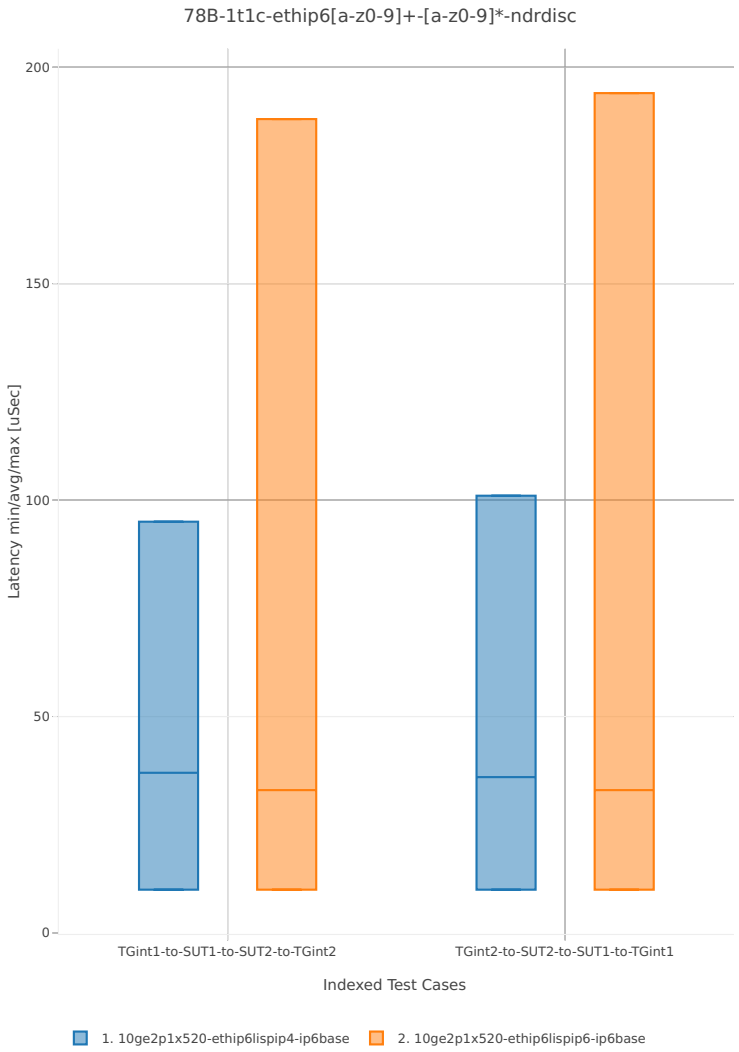


Figure 1. VPP 1thread 1core - packet latency for Phy-to-Phy IPv6 Overlay Tunnels.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/ip6_tunnels
$ grep -E "78B-1t1c-ethip6[a-z0-9]+-[a-z0-9]*-ndrdisc" *
```

VPP packet latency in 2t2c setup (2thread, 2core) is presented in the graph below.

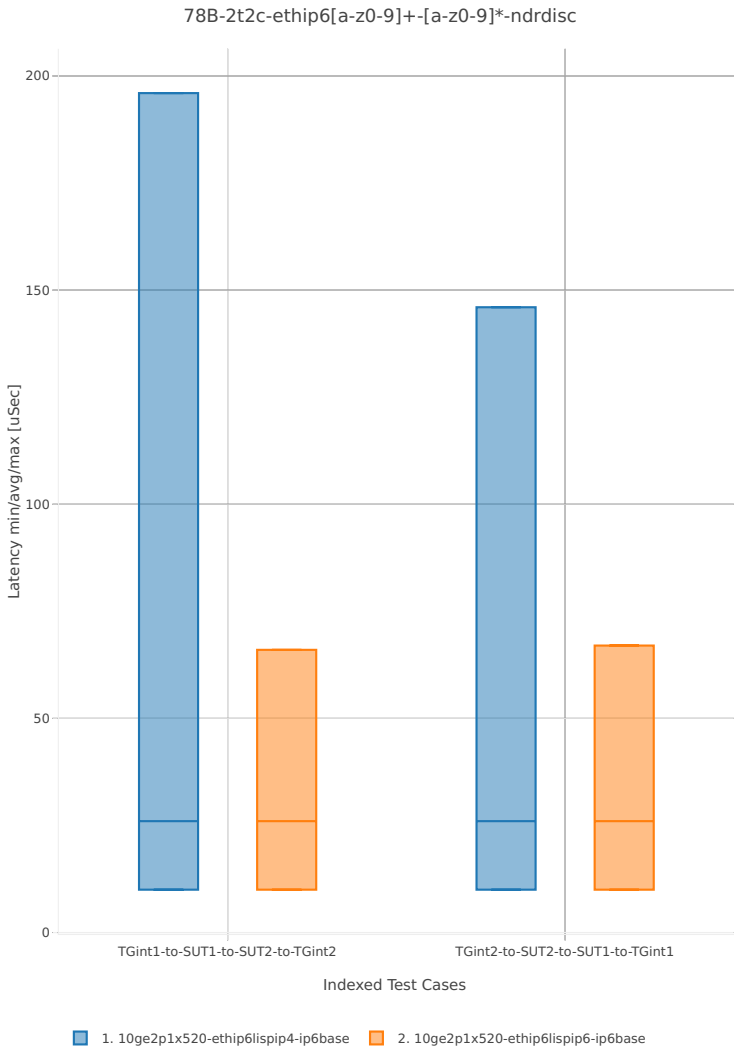


Figure 2. VPP 2threads 2cores - packet latency for Phy-to-Phy IPv6 Overlay Tunnels.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/ip6_tunnels
$ grep -E "78B-2t2c-ethip6[a-z0-9]+-[a-z0-9]*-ndrdisc" *
```

2.5.7 VM vhost Connections

This section includes summary graphs of VPP Phy-to-VM(s)-to-Phy packet latency with with VM virtio and VPP vhost-user virtual interfaces measured at 50% of discovered NDR throughput rate. Latency is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

VPP packet latency in 1t1c setup (1thread, 1core) is presented in the graph below.

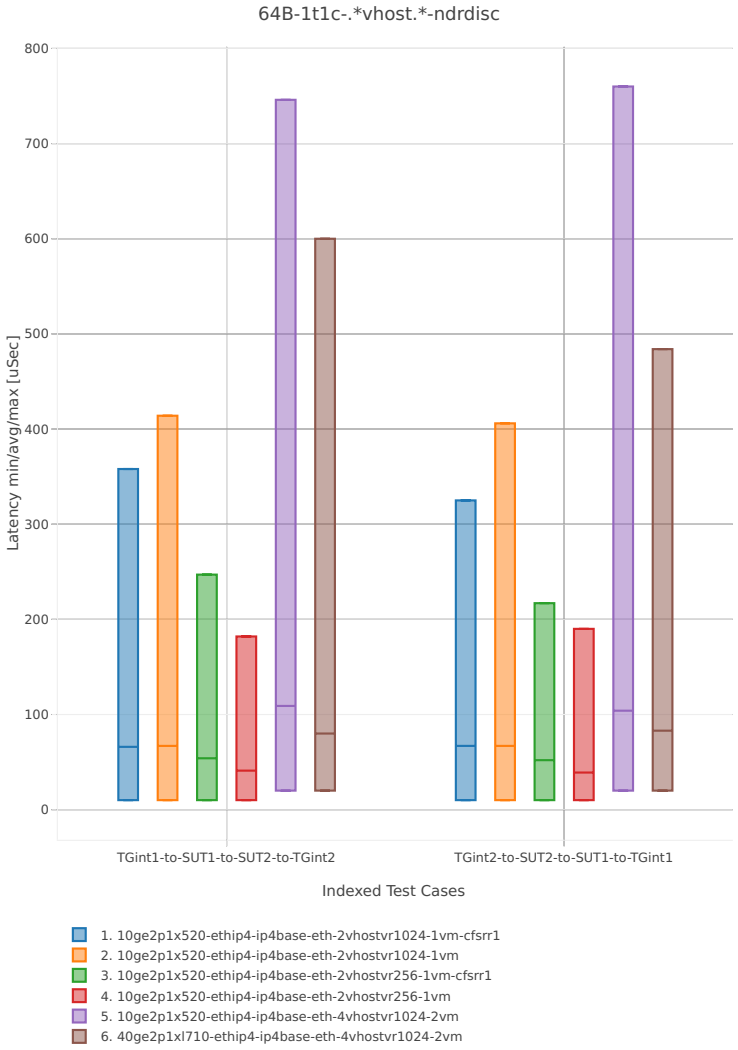


Figure 1a. VPP 1thread 1core - packet latency for Phy-to-VM-to-Phy VM vhost-user selected TCs (ipv4).

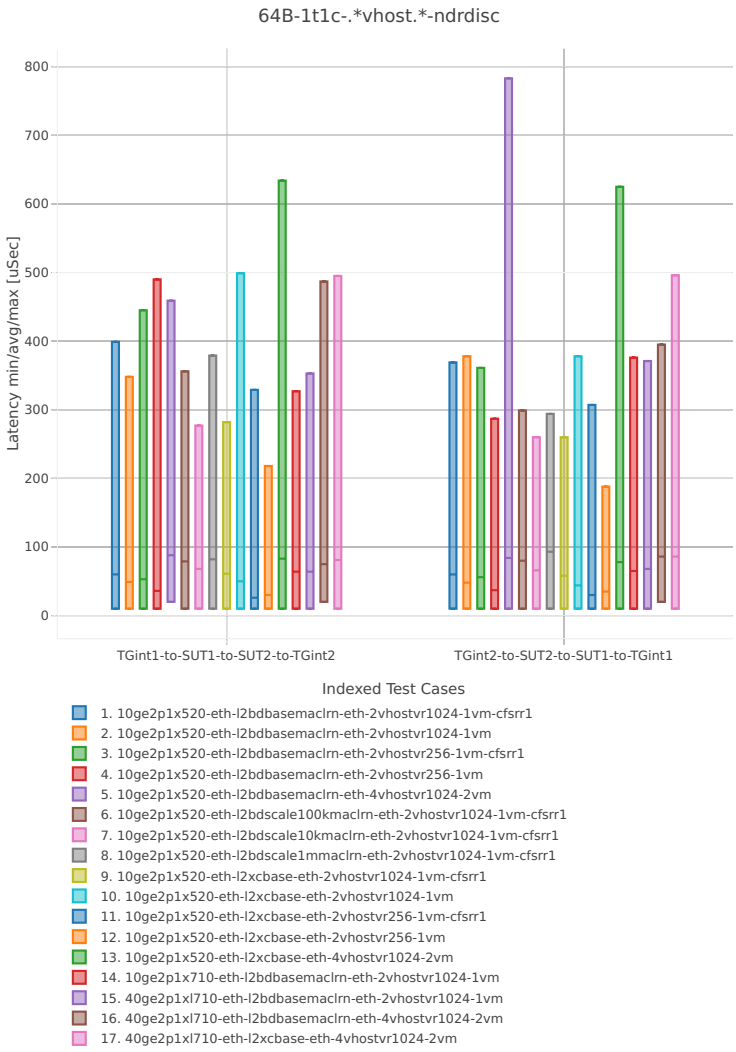


Figure 1b. VPP 1thread 1core - packet latency for Phy-to-VM-to-Phy VM vhost-user selected TCs (l2).

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/vm_vhost
$ grep -E "64B-1t1c-.*vhost.*-ndrdisc" *
```

VPP packet latency in 2t2c setup (2thread, 2core) is presented in the graph below.

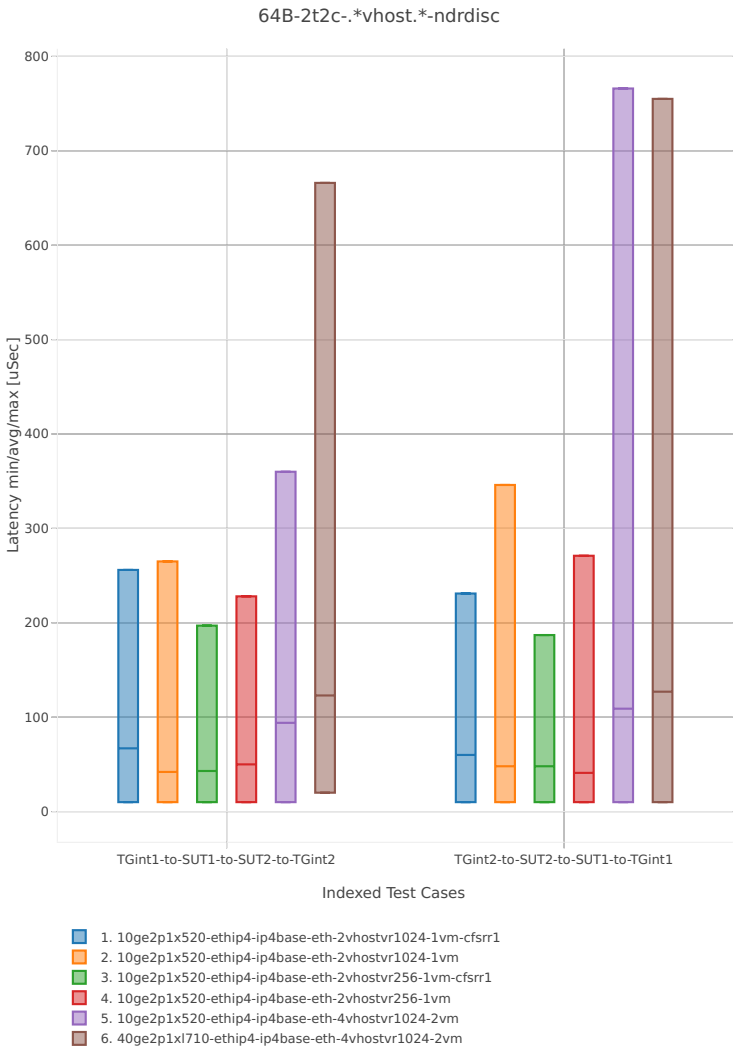


Figure 2a. VPP 2threads 2cores - packet latency for Phy-to-VM-to-Phy VM vhost-user selected TCs (ipv4).

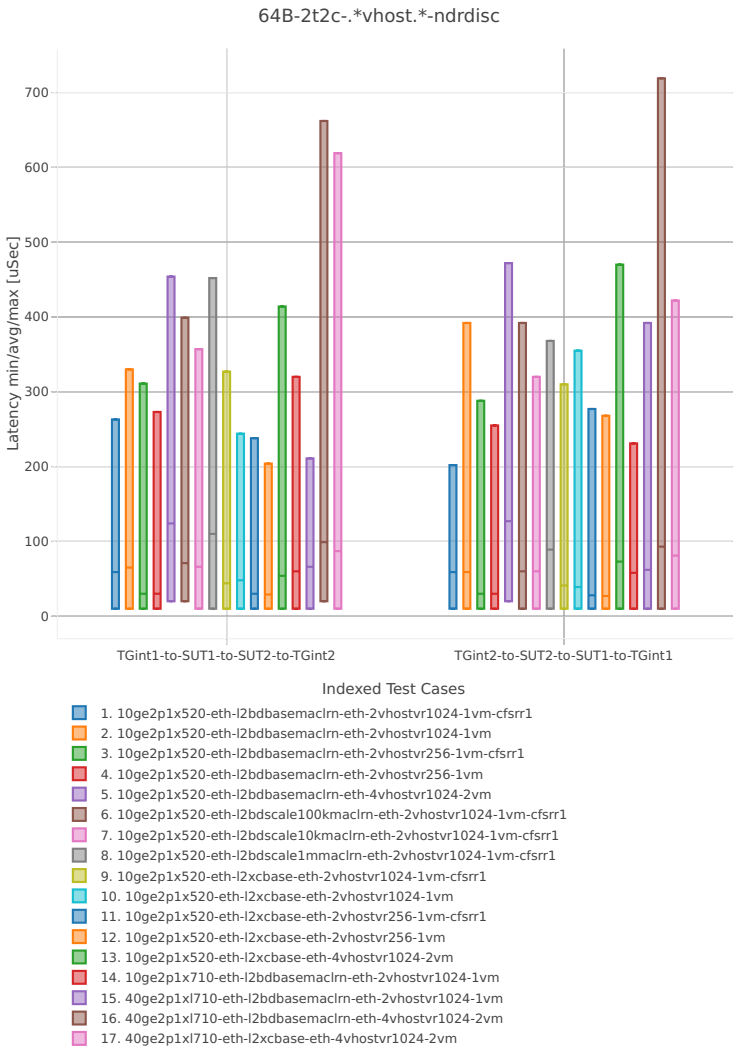


Figure 2b. VPP 2threads 2cores - packet latency for Phy-to-VM-to-Phy VM vhost-user selected TCs (l2).

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/vm_vhost
$ grep -E "64B-2t2c-.*vhost.*-ndrdisc" *
```

2.5.8 Container memif Connections

This section includes summary graphs of VPP Phy-to-Phy packet latency with Container memif Connections measured at 50% of discovered NDR throughput rate. Latency is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

VPP packet latency in 1t1c setup (1thread, 1core) is presented in the graph below.

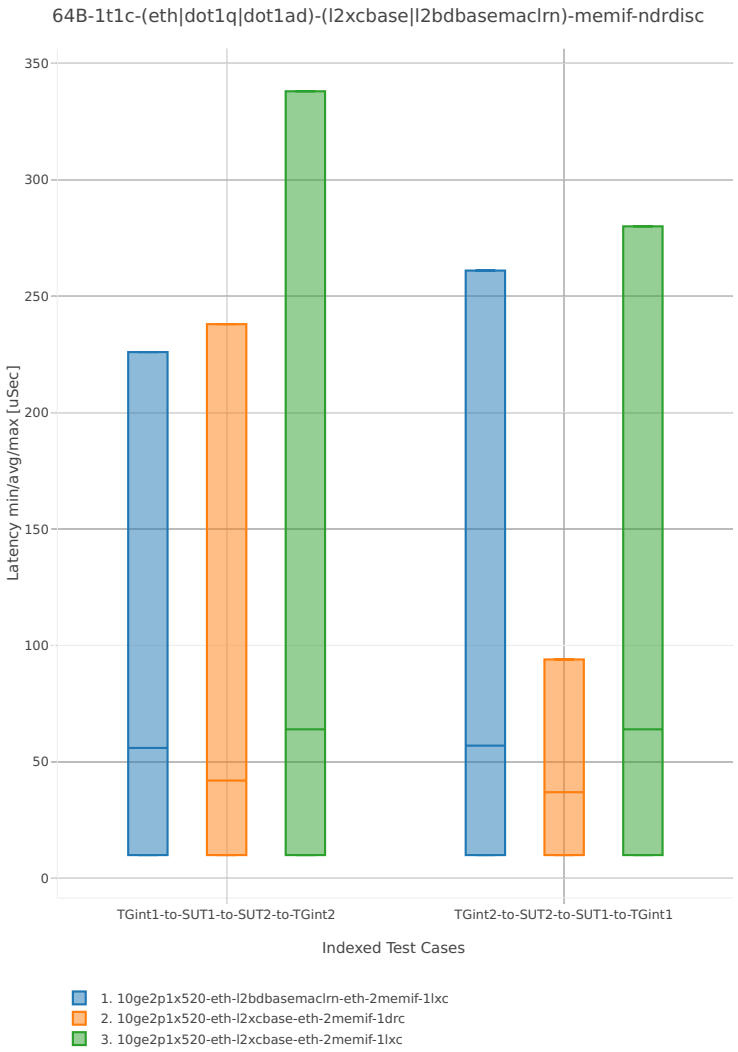


Figure 1. VPP 1thread 1core - packet latency for Phy-to-Phy L2 Ethernet Switching (base).

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/container_memif
$ grep -E "64B-1t1c-(eth|dot1q|dot1ad)-(l2xcbase|l2bdbasemaclrn)-.*ndrdisc" *
```

VPP packet latency in 2t2c setup (2thread, 2core) is presented in the graph below.

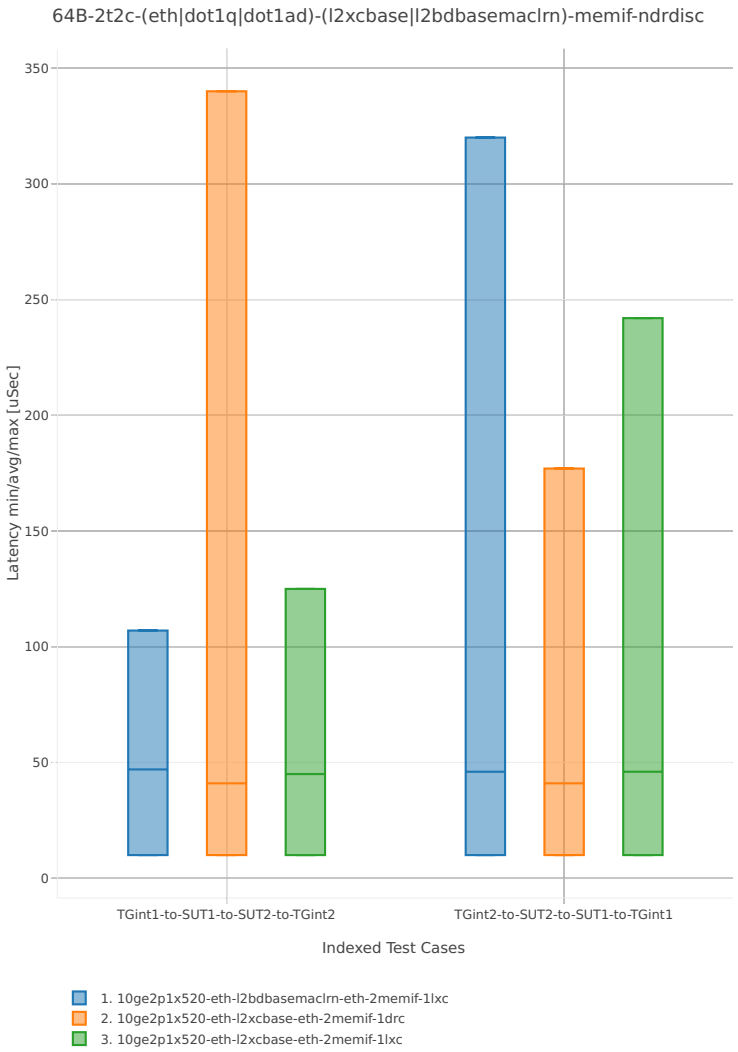


Figure 2. VPP 2threads 2cores - packet latency for Phy-to-Phy L2 Ethernet Switching (base).

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/container_memif
$ grep -E "64B-2t2c-(eth|dot1q|dot1ad)-(l2xcbase|l2bdbasemaclrn)-.*ndrdisc" *
```

2.5.9 Container Orchestrated Topologies

This section includes summary graphs of VPP Phy-to-Phy packet latency with CContainer Orchestrated Topologies measured at 50% of discovered NDR throughput rate. Latency is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

VPP packet latency in 1t1c setup (1thread, 1core) is presented in the graph below.

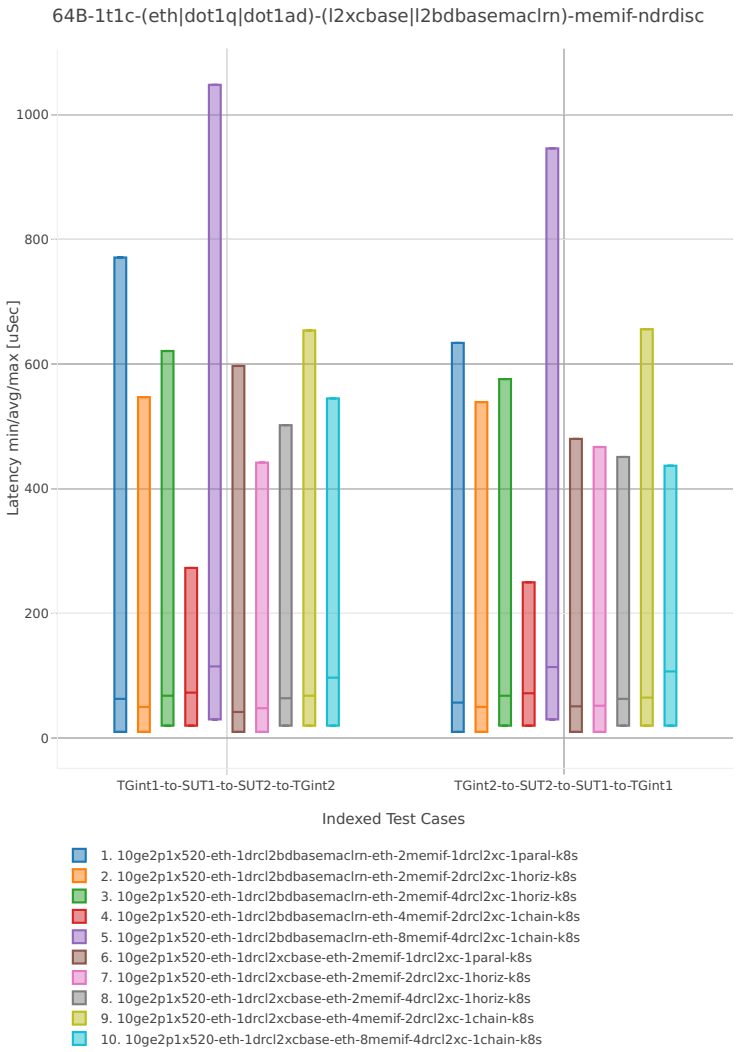


Figure 1. VPP 1thread 1core - packet latency for Phy-to-Phy L2 Ethernet Switching (base).

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/kubernetes/perf/container_memif
$ grep -E "64B-1t1c-(eth|dot1q|dot1ad)-[1-9]drc(l2xcbase|l2bdbasemaclrn)-.*ndrdisc" *
```

VPP packet latency in 2t2c setup (2thread, 2core) is presented in the graph below.

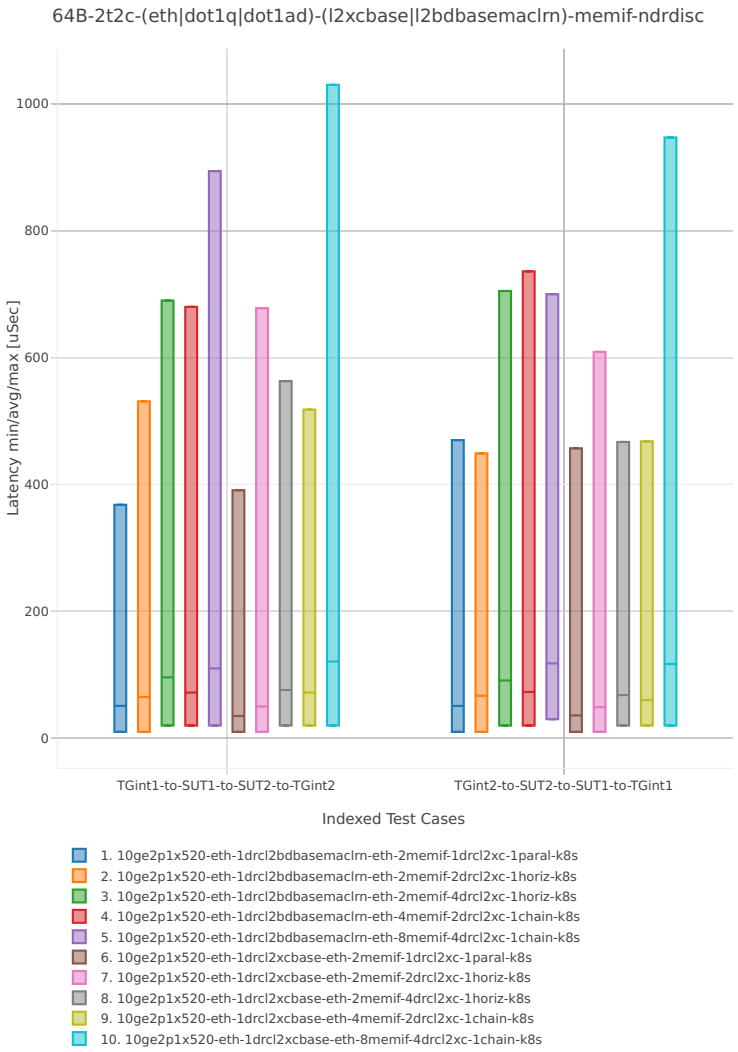


Figure 2. VPP 2threads 2cores - packet latency for Phy-to-Phy L2 Ethernet Switching (base).

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/kubernetes/perf/container_memif
$ grep -E "64B-2t2c-(eth|dot1q|dot1ad)-[1-9]drc(l2xcbase|l2bdbasemaclrn)-.*ndrdisc" *
```

2.5.10 IPSec Crypto HW: IP4 Routed-Forwarding

This section includes summary graphs of VPP Phy-to-Phy packet latency with IPSec encryption used in combination with IPv4 routed-forwarding, with latency measured at 50% of discovered NDR throughput rate. VPP IPSec encryption is accelerated using DPDK cryptodev library driving Intel Quick Assist (QAT) crypto PCIe hardware cards. Latency is reported for VPP running in multiple configurations of VPP worker thread(s), a.k.a. VPP data plane thread(s), and their physical CPU core(s) placement.

VPP packet latency in 1t1c setup (1thread, 1core) is presented in the graph below.

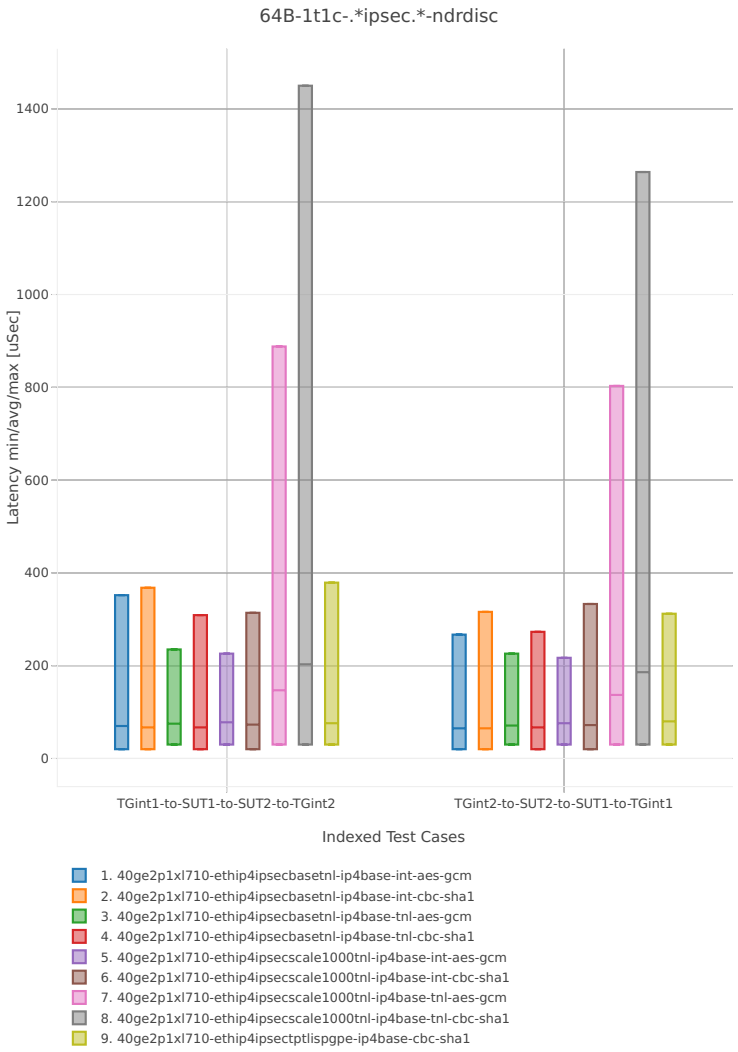


Figure 1. VPP 1thread 1core - packet latency for Phy-to-Phy IPSec HW with IPv4 Routed-Forwarding.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/crypto
$ grep -E "64B-1t1c-.*ipsec.*-ndrdisc" *
```

VPP packet latency in 2t2c setup (2thread, 2core) is presented in the graph below.

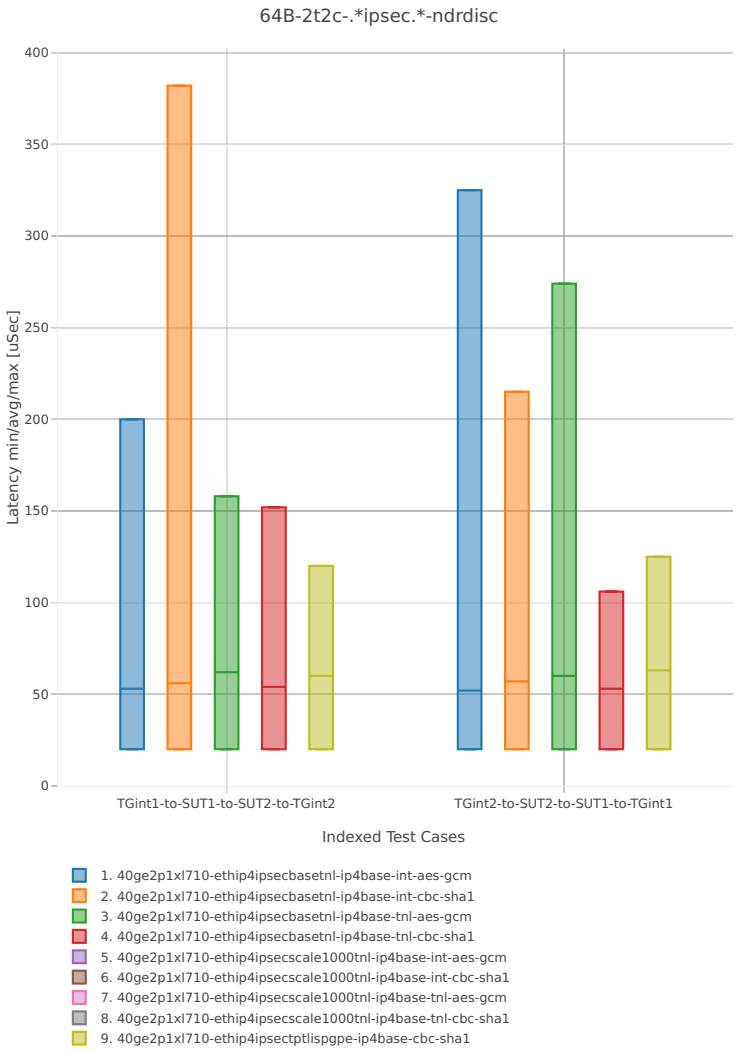


Figure 2. VPP 2threads 2cores - packet latency for Phy-to-Phy IPSec HW with IPv4 Routed-Forwarding.

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/crypto
$ grep -E "64B-2t2c-.*ipsec.*-ndrdisc" *
```

2.6 VPP HTTP Server Performance Results

Plotted results are generated by multiple executions of the same CSIT performance tests across three physical testbeds within LF FD.io labs. To provide a descriptive summary view, Box-and-Whisker plots are used to display variation in measured performance values, without making any assumptions of the underlying statistical distribution.

For each plotted test case, Box-and-Whisker plots show the quartiles (Min, 1st quartile / 25th percentile, 2nd quartile / 50th percentile / mean, 3rd quartile / 75th percentile, Max) across collected data set (data set size stated in the note below). Outliers are plotted as individual points. Min and max values are plotted as bottom and top Whiskers respectively. 2nd and 3rd quartiles are plotted as bottom and top edge of the box. If multiple samples match only two values, and all samples fall between them, then no whiskers are plotted. If all samples have the same value, only a horizontal line is plotted.

Note: Data sources for reported test results: i) FD.io test executor vpp performance jobs[61], ii) archived FD.io jobs test result output files.

2.6.1 Connections per second

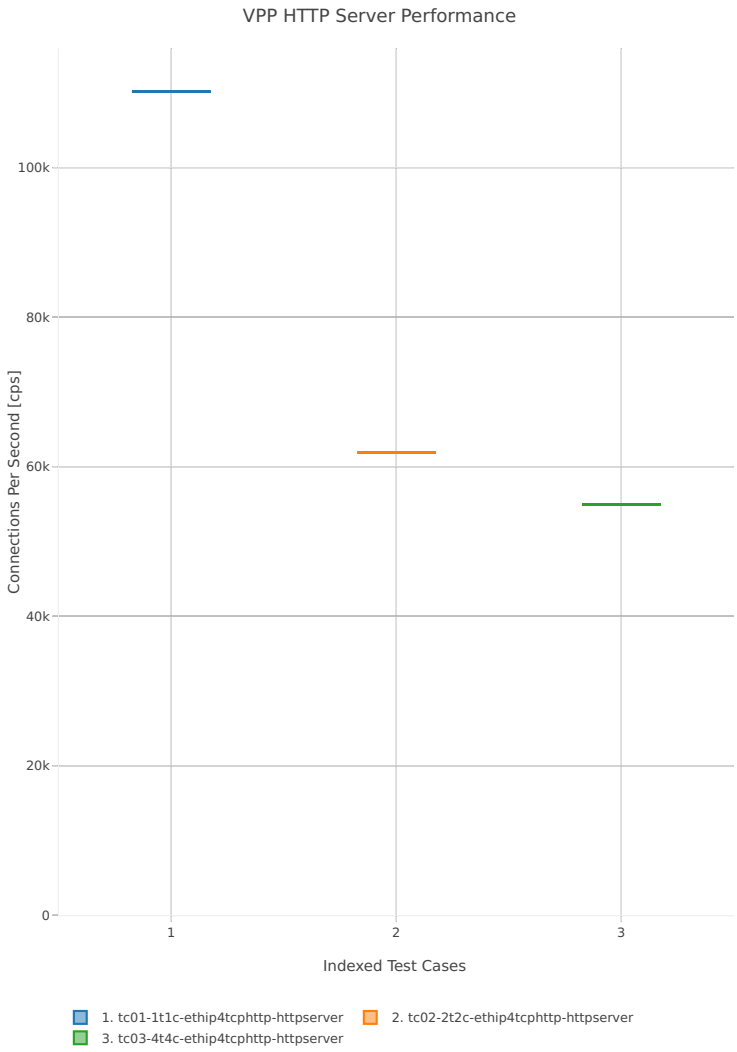


Figure 1. VPP HTTP Server Performance - Connections per Second

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/tcp
$ grep -HE '(1t1c|2t2c|4t4c)-ethip4tcphttp-httpserver-cps' *
```

[61] https://jenkins.fd.io/view/csit/job/csit-vpp-perf-1801-all

2.6.2 Requests per second

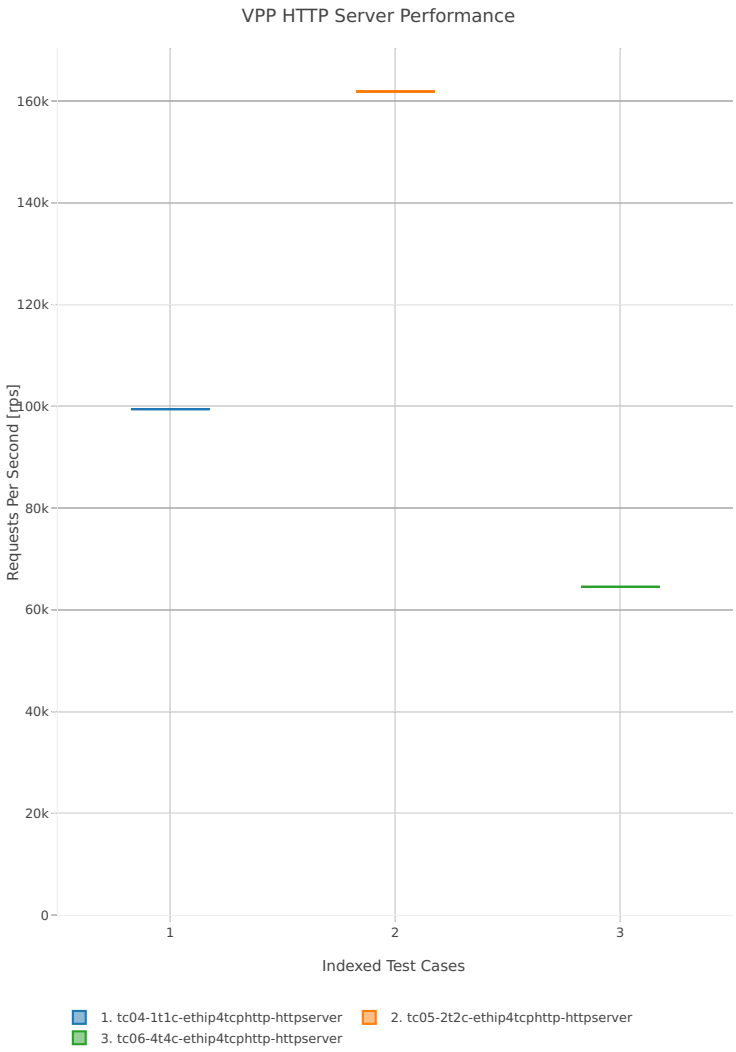


Figure 2. VPP HTTP Server Performance - Requests per Second

CSIT source code for the test cases used for above plots can be found in CSIT git repository:

```
$ cd tests/vpp/perf/tcp
$ grep -HE '(1t1c|2t2c|4t4c)-ethip4tcphttp-httpserver-rps' *
```

2.7 Test Environment

CSIT performance tests are executed on the three identical physical testbeds hosted by LF for FD.io project. Each testbed consists of two servers acting as Systems Under Test (SUT) and one server acting as Traffic Generator (TG).

2.7.1 Server Specification and Configuration

Complete specification and configuration of compute servers used in CSIT physical testbeds is maintained on wiki page CSIT LF Testbeds[62].

2.7.2 SUT Configuration

Host configuration

All hosts are Cisco UCS C240-M4 (2x Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz, 18c, 512GB RAM)

```
$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                36
On-line CPU(s) list:   0-35
Thread(s) per core:    1
Core(s) per socket:    18
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 63
Model name:            Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz
Stepping:              2
CPU MHz:               2294.249
BogoMIPS:              4589.82
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              46080K
NUMA node0 CPU(s):     0-17
NUMA node1 CPU(s):     18-35
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36␣
↪clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_
↪perfmon pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64␣
↪monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic movbe␣
↪popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm epb tpr_shadow vnmi flexpriority␣
↪ept vpid fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid cqm xsaveopt cqm_llc cqm_occup_llc␣
↪dtherm arat pln pts
```

BIOS settings

```
C240 /bios # show advanced detail
Set-up parameters:
    Intel(R) VT-d ATS Support: Enabled
    Adjacent Cache Line Prefetcher: Enabled
    All Onboard LOM Ports: Enabled
    Altitude: 300 M
    Bits per second: 115200
    Power Technology: Performance
    Channel Interleaving: Auto
    Intel(R) VT-d Coherency Support: Disabled
    Console Redirection: COM 0
    Number of Enabled Cores: All
    Energy Performance: Performance
    CPU Performance: Enterprise
    DCU IP Prefetcher: Enabled
```

(continues on next page)

[62] https://wiki.fd.io/view/CSIT/CSIT_LF_testbed

(continued from previous page)

```
    DCU Streamer Prefetch: Enabled
    Demand Scrub: Enabled
    Direct Cache Access Support: Auto
    Enhanced Intel Speedstep(R) Tec: Disabled
    Execute Disable: Enabled
    Flow Control: None
    Hardware Prefetcher: Enabled
    Intel(R) Hyper-Threading Techno: Disabled
    Intel(R) Turbo Boost Technology: Disabled
    Intel(R) VT: Enabled
    Intel(R) VT-d: Enabled
    Intel(R) Interrupt Remapping: Enabled
    Legacy USB Support: Enabled
    Extended APIC: XAPIC
    LOM Port 1 OptionROM: Enabled
    LOM Port 2 OptionROM: Enabled
    MMIO above 4GB: Enabled
    NUMA: Enabled
    PCI ROM CLP: Disabled
    Package C State Limit: C6 Retention
    Intel(R) Pass Through DMA: Disabled
    Patrol Scrub: Enabled
    xHCI Mode: Disabled
    All PCIe Slots OptionROM: Enabled
    PCIe Slot:1 OptionROM: Disabled
    PCIe Slot:2 OptionROM: Disabled
    PCIe Slot:3 OptionROM: Disabled
    PCIe Slot:4 OptionROM: Disabled
    PCIe Slot:5 OptionROM: Disabled
    PCIe Slot:6 OptionROM: Disabled
    PCIe Slot:HBA Link Speed: GEN3
    PCIe Slot:HBA OptionROM: Enabled
    PCIe Slot:MLOM OptionROM: Enabled
    PCIe Slot:N1 OptionROM: Enabled
    PCIe Slot:N2 OptionROM: Enabled
    Processor Power state C1 Enhanc: Disabled
    Processor C3 Report: Disabled
    Processor C6 Report: Disabled
    P-STATE Coordination: HW ALL
    Putty KeyPad: ESCN
    Energy Performance Tuning: BIOS
    QPI Link Frequency Select: Auto
    QPI Snoop Mode: Home Snoop
    Rank Interleaving: Auto
    Redirection After BIOS POST: Always Enable
    PCH SATA Mode: AHCI
    Select Memory RAS: Maximum Performance
    SR-IOV Support: Enabled
    Terminal Type: VT100
    Port 60/64 Emulation: Enabled
    Workload Configuration: Balanced
    CDN Support for VIC: Disabled
    Out-of-Band Management: Disabled
```

NIC models and placement

In addition to CIMC and Management, each TG has 4x Intel X710 10GB NIC (=8 ports) and 2x Intel XL710 40GB NIC (=4 ports), whereas each SUT has:

- 1x Intel X520 NIC (10GB, 2 ports),
- 1x Cisco VIC 1385 (40GB, 2 ports),

- 1x Intel XL710 NIC (40GB, 2 ports),
- 1x Intel X710 NIC (10GB, 2 ports),
- 1x Cisco VIC 1227 (10GB, 2 ports).

This allows for a total of five ring topologies, each using ports on specific NIC model, enabling per NIC model benchmarking.

- 0a:00.0 Ethernet controller: Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection (rev 01) Subsystem: Intel Corporation Ethernet Server Adapter X520-2
- 0a:00.1 Ethernet controller: Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection (rev 01) Subsystem: Intel Corporation Ethernet Server Adapter X520-2
- 06:00.0 Ethernet controller: Cisco Systems Inc VIC Ethernet NIC (rev a2) Subsystem: Cisco Systems Inc VIC 1227 PCIe Ethernet NIC
- 07:00.0 Ethernet controller: Cisco Systems Inc VIC Ethernet NIC (rev a2) Subsystem: Cisco Systems Inc VIC 1227 PCIe Ethernet NIC
- 13:00.0 Ethernet controller: Cisco Systems Inc VIC Ethernet NIC (rev a2) Subsystem: Cisco Systems Inc VIC 1385 PCIe Ethernet NIC
- 15:00.0 Ethernet controller: Cisco Systems Inc VIC Ethernet NIC (rev a2) Subsystem: Cisco Systems Inc VIC 1385 PCIe Ethernet NIC
- 85:00.0 Ethernet controller: Intel Corporation Ethernet Controller XL710 for 40GbE QSFP+ (rev 01) Subsystem: Intel Corporation Ethernet Converged Network Adapter XL710-Q2
- 85:00.1 Ethernet controller: Intel Corporation Ethernet Controller XL710 for 40GbE QSFP+ (rev 01) Subsystem: Intel Corporation Ethernet Converged Network Adapter XL710-Q2
- 87:00.0 Ethernet controller: Intel Corporation Ethernet Controller X710 for 10GbE SFP+ (rev 01) Subsystem: Intel Corporation Ethernet Converged Network Adapter X710-2
- 87:00.1 Ethernet controller: Intel Corporation Ethernet Controller X710 for 10GbE SFP+ (rev 01) Subsystem: Intel Corporation Ethernet Converged Network Adapter X710-2

2.7.3 SUT Configuration - Host OS Linux

Software details (OS, configuration) of physical testbeds are maintained on wiki page CSIT LF Testbeds[63].

System provisioning is done by combination of PXE boot unattented install and Ansible[64] described in CSIT Testbed Setup[65].

Below a subset of the running configuration:

```
$ lsb_release -a
No LSB modules are available.
Distributor ID:     Ubuntu
Description:        Ubuntu 16.04.1 LTS
Release:    16.04
Codename:   xenial
```

```
$ cat /sys/devices/system/node/node*/meminfo
Node 0 MemTotal:       264048168 kB
Node 0 MemFree:        257730716 kB
Node 0 MemUsed:          6317452 kB
Node 0 Active:           1079920 kB
Node 0 Inactive:          470064 kB
```

(continues on next page)

[63] https://wiki.fd.io/view/CSIT/CSIT_LF_testbed
[64] https://www.ansible.com
[65] https://git.fd.io/csit/tree/resources/tools/testbed-setup/README.md?h=rls1801_2

(continued from previous page)

```
Node 0 Active(anon):     674772 kB
Node 0 Inactive(anon):   248572 kB
Node 0 Active(file):     405148 kB
Node 0 Inactive(file):   221492 kB
Node 0 Unevictable:           0 kB
Node 0 Mlocked:               0 kB
Node 0 Dirty:                12 kB
Node 0 Writeback:             0 kB
Node 0 FilePages:       1270432 kB
Node 0 Mapped:            20116 kB
Node 0 AnonPages:        279548 kB
Node 0 Shmem:            643796 kB
Node 0 KernelStack:        3376 kB
Node 0 PageTables:         1316 kB
Node 0 NFS_Unstable:          0 kB
Node 0 Bounce:                0 kB
Node 0 WritebackTmp:          0 kB
Node 0 Slab:              80428 kB
Node 0 SReclaimable:      38288 kB
Node 0 SUnreclaim:        42140 kB
Node 0 AnonHugePages:    270336 kB
Node 0 HugePages_Total:  2048
Node 0 HugePages_Free:   2048
Node 0 HugePages_Surp:      0
Node 1 MemTotal:       264237596 kB
Node 1 MemFree:        256758976 kB
Node 1 MemUsed:         7478620 kB
Node 1 Active:          1746052 kB
Node 1 Inactive:         981104 kB
Node 1 Active(anon):    1272936 kB
Node 1 Inactive(anon):   849968 kB
Node 1 Active(file):     473116 kB
Node 1 Inactive(file):   131136 kB
Node 1 Unevictable:           0 kB
Node 1 Mlocked:               0 kB
Node 1 Dirty:                 0 kB
Node 1 Writeback:             0 kB
Node 1 FilePages:       2715284 kB
Node 1 Mapped:            75928 kB
Node 1 AnonPages:         11920 kB
Node 1 Shmem:           2111036 kB
Node 1 KernelStack:        2576 kB
Node 1 PageTables:         1348 kB
Node 1 NFS_Unstable:          0 kB
Node 1 Bounce:                0 kB
Node 1 WritebackTmp:          0 kB
Node 1 Slab:              90604 kB
Node 1 SReclaimable:      55384 kB
Node 1 SUnreclaim:        35220 kB
Node 1 AnonHugePages:      6144 kB
Node 1 HugePages_Total:  2048
Node 1 HugePages_Free:   2048
Node 1 HugePages_Surp:      0
```

Kernel boot parameters used in CSIT performance testbeds

- **isolcpus=<cpu number>-<cpu number>** used for all cpu cores apart from first core of each socket used for running VPP worker threads and Qemu/LXC processes https://www.kernel.org/doc/Documentation/admin-guide/kernel-parameters.txt
- **intel_pstate=disable** - [X86] Do not enable intel_pstate as the default scaling driver for the supported processors. Intel P-State driver decide what P-state (CPU core power state) to use based

on requesting policy from the cpufreq core. [X86 - Either 32-bit or 64-bit x86] https://www.kernel.org/doc/Documentation/cpu-freq/intel-pstate.txt

- **nohz_full=<cpu number>-<cpu number>** - [KNL,BOOT] In kernels built with CONFIG_NO_HZ_FULL=y, set the specified list of CPUs whose tick will be stopped whenever possible. The boot CPU will be forced outside the range to maintain the timekeeping. The CPUs in this range must also be included in the rcu_nocbs= set. Specifies the adaptive-ticks CPU cores, causing kernel to avoid sending scheduling-clock interrupts to listed cores as long as they have a single runnable task. [KNL - Is a kernel start-up parameter, SMP - The kernel is an SMP kernel]. https://www.kernel.org/doc/Documentation/timers/NO_HZ.txt
- **rcu_nocbs** - [KNL] In kernels built with CONFIG_RCU_NOCB_CPU=y, set the specified list of CPUs to be no-callback CPUs, that never queue RCU callbacks (read-copy update). https://www.kernel.org/doc/Documentation/admin-guide/kernel-parameters.txt

Applied command line boot parameters:

```
$ cat /proc/cmdline
BOOT_IMAGE=/vmlinuz-4.4.0-72-generic root=UUID=35ea11e4-e44f-4f67-8cbe-12f09c49ed90 ro isolcpus=1-
↪17,19-35 nohz_full=1-17,19-35 rcu_nocbs=1-17,19-35 intel_pstate=disable console=tty0␣
↪console=ttyS0,115200n8
```

Mount listing

```
$ cat /proc/mounts
sysfs /sys sysfs rw,nosuid,nodev,noexec,relatime 0 0
proc /proc proc rw,nosuid,nodev,noexec,relatime 0 0
udev /dev devtmpfs rw,nosuid,relatime,size=264125468k,nr_inodes=66031367,mode=755 0 0
devpts /dev/pts devpts rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000 0 0
tmpfs /run tmpfs rw,nosuid,noexec,relatime,size=52828580k,mode=755 0 0
/dev/sda2 / ext4 rw,relatime,errors=remount-ro,data=ordered 0 0
securityfs /sys/kernel/security securityfs rw,nosuid,nodev,noexec,relatime 0 0
tmpfs /dev/shm tmpfs rw,nosuid,nodev 0 0
tmpfs /run/lock tmpfs rw,nosuid,nodev,noexec,relatime,size=5120k 0 0
tmpfs /sys/fs/cgroup tmpfs ro,nosuid,nodev,noexec,mode=755 0 0
cgroup /sys/fs/cgroup/systemd cgroup rw,nosuid,nodev,noexec,relatime,xattr,release_agent=/lib/
↪systemd/systemd-cgroups-agent,name=systemd 0 0
pstore /sys/fs/pstore pstore rw,nosuid,nodev,noexec,relatime 0 0
cgroup /sys/fs/cgroup/freezer cgroup rw,nosuid,nodev,noexec,relatime,freezer 0 0
cgroup /sys/fs/cgroup/net_cls,net_prio cgroup rw,nosuid,nodev,noexec,relatime,net_cls,net_prio 0 0
cgroup /sys/fs/cgroup/cpu,cpuacct cgroup rw,nosuid,nodev,noexec,relatime,cpu,cpuacct 0 0
cgroup /sys/fs/cgroup/memory cgroup rw,nosuid,nodev,noexec,relatime,memory 0 0
cgroup /sys/fs/cgroup/blkio cgroup rw,nosuid,nodev,noexec,relatime,blkio 0 0
cgroup /sys/fs/cgroup/perf_event cgroup rw,nosuid,nodev,noexec,relatime,perf_event 0 0
cgroup /sys/fs/cgroup/devices cgroup rw,nosuid,nodev,noexec,relatime,devices 0 0
cgroup /sys/fs/cgroup/cpuset cgroup rw,nosuid,nodev,noexec,relatime,cpuset,clone_children 0 0
cgroup /sys/fs/cgroup/hugetlb cgroup rw,nosuid,nodev,noexec,relatime,hugetlb 0 0
cgroup /sys/fs/cgroup/pids cgroup rw,nosuid,nodev,noexec,relatime,pids 0 0
systemd-1 /proc/sys/fs/binfmt_misc autofs rw,relatime,fd=26,pgrp=1,timeout=0,minproto=5,maxproto=5,
↪direct 0 0
hugetlbfs /dev/hugepages hugetlbfs rw,relatime 0 0
debugfs /sys/kernel/debug debugfs rw,relatime 0 0
mqueue /dev/mqueue mqueue rw,relatime 0 0
tracefs /sys/kernel/debug/tracing tracefs rw,relatime 0 0
fusectl /sys/fs/fuse/connections fusectl rw,relatime 0 0
/dev/sda1 /boot ext4 rw,relatime,data=ordered 0 0
none /mnt/huge hugetlbfs rw,relatime,pagesize=2048k 0 0
lxcfs /var/lib/lxcfs fuse.lxcfs rw,nosuid,nodev,relatime,user_id=0,group_id=0,allow_other 0 0
```

Package listing

```
$ dpkg -l | grep '^ii' | awk '{print $2 ": " $3}'
accountsservice: 0.6.40-2ubuntu11.1
```

(continues on next page)

(continued from previous page)

```
acl: 2.2.52-3
adduser: 3.113+nmu3ubuntu4
apparmor: 2.10.95-0ubuntu2.6
apt: 1.2.12~ubuntu16.04.1
apt-transport-https: 1.2.24
apt-utils: 1.2.12~ubuntu16.04.1
aufs-tools: 1:3.2+20130722-1.1ubuntu1
autoconf: 2.69-9
automake: 1:1.15-4ubuntu1
autotools-dev: 20150820.1
base-files: 9.4ubuntu4.2
base-passwd: 3.5.39
bash: 4.3-14ubuntu1.1
binutils: 2.26.1-1ubuntu1~16.04.3
bridge-utils: 1.5-9ubuntu1
bsdutils: 1:2.27.1-6ubuntu3.1
build-essential: 12.1ubuntu2
busybox-initramfs: 1:1.22.0-15ubuntu1
busybox-static: 1:1.22.0-15ubuntu1
bzip2: 1.0.6-8
busybox-static: 1:1.22.0-15ubuntu1
bzip2: 1.0.6-8
ca-certificates: 20160104ubuntu1
ca-certificates-java: 20160321
cgroup-bin: 0.41-7ubuntu1
cgroup-lite: 1.11
cgroup-tools: 0.41-7ubuntu1
cloud-image-utils: 0.27-0ubuntu24
console-setup: 1.108ubuntu15.2
console-setup-linux: 1.108ubuntu15.2
corekeeper: 1.6
coreutils: 8.25-2ubuntu2
cpio: 2.11+dfsg-5ubuntu1
cpp: 4:5.3.1-1ubuntu1
cpp-5: 5.4.0-6ubuntu1~16.04.2
cpu-checker: 0.7-0ubuntu7
cpufrequtils: 008-1
crda: 3.13-1
cron: 3.0pl1-128ubuntu2
crudini: 0.7-1
dash: 0.5.8-2.1ubuntu2
dbus: 1.10.6-1ubuntu3
debconf: 1.5.58ubuntu1
debconf-i18n: 1.5.58ubuntu1
debianutils: 4.7
debootstrap: 1.0.78+nmu1ubuntu1.3
dh-python: 2.20151103ubuntu1.1
diffutils: 1:3.3-3
distro-info: 0.14build1
distro-info-data: 0.28ubuntu0.1
dkms: 2.2.0.3-2ubuntu11.2
dmidecode: 3.0-2ubuntu0.1
dns-root-data: 2015052300+h+1
dnsmasq-base: 2.75-1ubuntu0.16.04.2
docker-ce: 17.09.0~ce-0~ubuntu
dpkg: 1.18.4ubuntu1.1
dpkg-dev: 1.18.4ubuntu1.1
e2fslibs:amd64: 1.42.13-1ubuntu1
e2fsprogs: 1.42.13-1ubuntu1
ebtables: 2.0.10.4-3.4ubuntu2
eject: 2.1.5+deb1+cvs20081104-13.1
```

(continues on next page)

(continued from previous page)

```
ethtool: 1:4.5-1
expect: 5.45-7
fakeroot: 1.20.2-1ubuntu1
file: 1:5.25-2ubuntu1
findutils: 4.6.0+git+20160126-2
fontconfig: 2.11.94-0ubuntu1.1
fontconfig-config: 2.11.94-0ubuntu1.1
fonts-dejavu-core: 2.35-1
fonts-dejavu-extra: 2.35-1
g++: 4:5.3.1-1ubuntu1
g++-5: 5.4.0-6ubuntu1~16.04.2
gcc: 4:5.3.1-1ubuntu1
gcc-5: 5.4.0-6ubuntu1~16.04.2
gcc-5-base:amd64: 5.4.0-6ubuntu1~16.04.2
gcc-6-base:amd64: 6.0.1-0ubuntu1
gdb: 7.11.1-0ubuntu1~16.5
gdbserver: 7.11.1-0ubuntu1~16.5
genisoimage: 9:1.1.11-3ubuntu1
gettext-base: 0.19.7-2ubuntu3
gir1.2-glib-2.0:amd64: 1.46.0-3ubuntu1
git: 1:2.7.4-0ubuntu1
git-man: 1:2.7.4-0ubuntu1
gnupg: 1.4.20-1ubuntu3.1
gpgv: 1.4.20-1ubuntu3.1
grep: 2.25-1~16.04.1
grub-common: 2.02~beta2-36ubuntu3.1
grub-gfxpayload-lists: 0.7
grub-pc: 2.02~beta2-36ubuntu3.1
grub-pc-bin: 2.02~beta2-36ubuntu3.1
grub2-common: 2.02~beta2-36ubuntu3.1
gzip: 1.6-4ubuntu1
hicolor-icon-theme: 0.15-0ubuntu1
hostname: 3.16ubuntu2
ifupdown: 0.8.10ubuntu1
init: 1.29ubuntu2
init-system-helpers: 1.29ubuntu2
initramfs-tools: 0.122ubuntu8.1
initramfs-tools-bin: 0.122ubuntu8.1
initramfs-tools-core: 0.122ubuntu8.1
initscripts: 2.88dsf-59.3ubuntu2
insserv: 1.14.0-5ubuntu3
installation-report: 2.60ubuntu1
iproute2: 4.3.0-1ubuntu3
iptables: 1.6.0-2ubuntu3
iputils-ping: 3:20121221-5ubuntu2
ipxe-qemu: 1.0.0+git-20150424.a25a16d-1ubuntu1
isc-dhcp-client: 4.3.3-5ubuntu12.1
isc-dhcp-common: 4.3.3-5ubuntu12.1
iso-codes: 3.65-1
iw: 3.17-1
java-common: 0.56ubuntu2
kbd: 1.15.5-1ubuntu4
keyboard-configuration: 1.108ubuntu15.2
klibc-utils: 2.0.4-8ubuntu1.16.04.1
kmod: 22-1ubuntu4
krb5-locales: 1.13.2+dfsg-5
kubeadm: 1.8.4-00
kubectl: 1.8.4-00
kubelet: 1.8.4-00
kubernetes-cni: 0.5.1-00
language-selector-common: 0.165.3
```

(continues on next page)

(continued from previous page)

```
laptop-detect: 0.13.7ubuntu2
less: 481-2.1
libaccountsservice0:amd64: 0.6.40-2ubuntu11.1
libacl1:amd64: 2.2.52-3
libaio1:amd64: 0.3.110-2
libalgorithm-diff-perl: 1.19.03-1
libalgorithm-diff-xs-perl: 0.04-4build1
libalgorithm-merge-perl: 0.08-3
libapparmor-perl: 2.10.95-0ubuntu2.6
libapparmor1:amd64: 2.10.95-0ubuntu2
libapr1:amd64: 1.5.2-3
libapt-inst2.0:amd64: 1.2.12~ubuntu16.04.1
libapt-pkg5.0:amd64: 1.2.12~ubuntu16.04.1
libasan2:amd64: 5.4.0-6ubuntu1~16.04.2
libasn1-8-heimdal:amd64: 1.7~git20150920+dfsg-4ubuntu1
libasound2:amd64: 1.1.0-0ubuntu1
libasound2-data: 1.1.0-0ubuntu1
libasprintf0v5:amd64: 0.19.7-2ubuntu3
libasyncns0:amd64: 0.8-5build1
libatk1.0-0:amd64: 2.18.0-1
libatk1.0-data: 2.18.0-1
libatm1:amd64: 1:2.5.1-1.5
libatomic1:amd64: 5.4.0-6ubuntu1~16.04.2
libattr1:amd64: 1:2.4.47-2
libaudit-common: 1:2.4.5-1ubuntu2
libaudit1:amd64: 1:2.4.5-1ubuntu2
libavahi-client3:amd64: 0.6.32~rc+dfsg-1ubuntu2
libavahi-common-data:amd64: 0.6.32~rc+dfsg-1ubuntu2
libavahi-common3:amd64: 0.6.32~rc+dfsg-1ubuntu2
libbabeltrace-ctf1:amd64: 1.3.2-1
libbabeltrace1:amd64: 1.3.2-1
libblkid1:amd64: 2.27.1-6ubuntu3.1
libbluetooth3:amd64: 5.37-0ubuntu5
libboost-iostreams1.58.0:amd64: 1.58.0+dfsg-5ubuntu3.1
libboost-random1.58.0:amd64: 1.58.0+dfsg-5ubuntu3.1
libboost-system1.58.0:amd64: 1.58.0+dfsg-5ubuntu3.1
libboost-thread1.58.0:amd64: 1.58.0+dfsg-5ubuntu3.1
libbrlapi0.6:amd64: 5.3.1-2ubuntu2.1
libbsd0:amd64: 0.8.2-1
libbz2-1.0:amd64: 1.0.6-8
libc-bin: 2.23-0ubuntu3
libc-dev-bin: 2.23-0ubuntu9
libc6-dbg:amd64: 2.23-0ubuntu9
libc6-dev:amd64: 2.23-0ubuntu9
libcaca0:amd64: 0.99.beta19-2build2~gcc5.2
libcacard0:amd64: 1:2.5.0-2
libcairo2:amd64: 1.14.6-1
libcap-ng0:amd64: 0.7.7-1
libcap2:amd64: 1:2.24-12
libcap2-bin: 1:2.24-12
libcc1-0:amd64: 5.4.0-6ubuntu1~16.04.2
libcgroup1:amd64: 0.41-7ubuntu1
libcilkrts5:amd64: 5.4.0-6ubuntu1~16.04.2
libcomerr2:amd64: 1.42.13-1ubuntu1
libcpufreq0: 008-1
libcryptsetup4:amd64: 2:1.6.6-5ubuntu2
libcups2:amd64: 2.1.3-4
libcurl3-gnutls:amd64: 7.47.0-1ubuntu2.1
libdatrie1:amd64: 0.2.10-2
libdb5.3:amd64: 5.3.28-11
libdbus-1-3:amd64: 1.10.6-1ubuntu3
```

(continues on next page)

(continued from previous page)

```
libdbus-glib-1-2:amd64: 0.106-1
libdebconfclient0:amd64: 0.198ubuntu1
libdevmapper1.02.1:amd64: 2:1.02.110-1ubuntu10
libdns-export162: 1:9.10.3.dfsg.P4-8ubuntu1.1
libdpkg-perl: 1.18.4ubuntu1.1
libdrm-amdgpu1:amd64: 2.4.67-1ubuntu0.16.04.2
libdrm-intel1:amd64: 2.4.67-1ubuntu0.16.04.2
libdrm-nouveau2:amd64: 2.4.67-1ubuntu0.16.04.2
libdrm-radeon1:amd64: 2.4.67-1ubuntu0.16.04.2
libdrm2:amd64: 2.4.67-1ubuntu0.16.04.2
libedit2:amd64: 3.1-20150325-1ubuntu2
libelf1:amd64: 0.165-3ubuntu1
liberror-perl: 0.17-1.2
libestr0: 0.1.10-1
libexpat1:amd64: 2.1.0-7ubuntu0.16.04.2
libexpat1-dev:amd64: 2.1.0-7ubuntu0.16.04.2
libfakeroot:amd64: 1.20.2-1ubuntu1
libfdisk1:amd64: 2.27.1-6ubuntu3.1
libfdt1:amd64: 1.4.0+dfsg-2
libffi6:amd64: 3.2.1-4
libfile-fcntllock-perl: 0.22-3
libflac8:amd64: 1.3.1-4
libfontconfig1:amd64: 2.11.94-0ubuntu1.1
libfontenc1:amd64: 1:1.1.3-1
libfreetype6:amd64: 2.6.1-0.1ubuntu2
libfribidi0:amd64: 0.19.7-1
libfuse2:amd64: 2.9.4-1ubuntu3
libgcc-5-dev:amd64: 5.4.0-6ubuntu1~16.04.2
libgcc1:amd64: 1:6.0.1-0ubuntu1
libgcrypt20:amd64: 1.6.5-2ubuntu0.2
libgdbm3:amd64: 1.8.3-13.1
libgdk-pixbuf2.0-0:amd64: 2.32.2-1ubuntu1.2
libgdk-pixbuf2.0-common: 2.32.2-1ubuntu1.2
libgif7:amd64: 5.1.4-0.3~16.04
libgirepository-1.0-1:amd64: 1.46.0-3ubuntu1
libgl1-mesa-dri:amd64: 11.2.0-1ubuntu2.2
libgl1-mesa-glx:amd64: 11.2.0-1ubuntu2.2
libglapi-mesa:amd64: 11.2.0-1ubuntu2.2
libglib2.0-0:amd64: 2.48.1-1~ubuntu16.04.1
libglib2.0-bin: 2.48.1-1~ubuntu16.04.1
libglib2.0-data: 2.48.1-1~ubuntu16.04.1
libglib2.0-dev: 2.48.1-1~ubuntu16.04.1
libgmp10:amd64: 2:6.1.0+dfsg-2
libgnutls-openssl27:amd64: 3.4.10-4ubuntu1.1
libgnutls30:amd64: 3.4.10-4ubuntu1.1
libgomp1:amd64: 5.4.0-6ubuntu1~16.04.2
libgpg-error0:amd64: 1.21-2ubuntu1
libgraphite2-3:amd64: 1.3.6-1ubuntu1
libgssapi-krb5-2:amd64: 1.13.2+dfsg-5
libgssapi3-heimdal:amd64: 1.7~git20150920+dfsg-4ubuntu1
libgtk2.0-0:amd64: 2.24.30-1ubuntu1.16.04.2
libgtk2.0-bin: 2.24.30-1ubuntu1.16.04.2
libgtk2.0-common: 2.24.30-1ubuntu1.16.04.2
libharfbuzz0b:amd64: 1.0.1-1ubuntu0.1
libhcrypto4-heimdal:amd64: 1.7~git20150920+dfsg-4ubuntu1
libheimbase1-heimdal:amd64: 1.7~git20150920+dfsg-4ubuntu1
libheimntlm0-heimdal:amd64: 1.7~git20150920+dfsg-4ubuntu1
libhogweed4:amd64: 3.2-1
libhx509-5-heimdal:amd64: 1.7~git20150920+dfsg-4ubuntu1
libice-dev:amd64: 2:1.0.9-1
libice6:amd64: 2:1.0.9-1
```

(continues on next page)

(continued from previous page)

```
libicu55:amd64: 55.1-7
libidn11:amd64: 1.32-3ubuntu1.1
libisc-export160: 1:9.10.3.dfsg.P4-8ubuntu1.1
libiscsi2:amd64: 1.12.0-2
libisl15:amd64: 0.16.1-1
libitm1:amd64: 5.4.0-6ubuntu1~16.04.2
libjbig0:amd64: 2.1-3.1
libjpeg-turbo8:amd64: 1.4.2-0ubuntu3
libjpeg8:amd64: 8c-2ubuntu8
libjson-c2:amd64: 0.11-4ubuntu2
libk5crypto3:amd64: 1.13.2+dfsg-5
libkeyutils1:amd64: 1.5.9-8ubuntu1
libklibc: 2.0.4-8ubuntu1.16.04.1
libkmod2:amd64: 22-1ubuntu4
libkrb5-26-heimdal:amd64: 1.7~git20150920+dfsg-4ubuntu1
libkrb5-3:amd64: 1.13.2+dfsg-5
libkrb5support0:amd64: 1.13.2+dfsg-5
liblcms2-2:amd64: 2.6-3ubuntu2
libldap-2.4-2:amd64: 2.4.42+dfsg-2ubuntu3.1
libllvm3.8:amd64: 1:3.8-2ubuntu4
liblocale-gettext-perl: 1.07-1build1
liblsan0:amd64: 5.4.0-6ubuntu1~16.04.2
libltdl-dev:amd64: 2.4.6-0.1
libltdl7:amd64: 2.4.6-0.1
liblxc1: 2.0.7-0ubuntu1~16.04.2
liblz4-1:amd64: 0.0~r131-2ubuntu2
liblzma5:amd64: 5.1.1alpha+20120614-2ubuntu2
libmagic1:amd64: 1:5.25-2ubuntu1
libmnl0:amd64: 1.0.3-5
libmount1:amd64: 2.27.1-6ubuntu3.1
libmpc3:amd64: 1.0.3-1
libmpdec2:amd64: 2.4.2-1
libmpfr4:amd64: 3.1.4-1
libmpx0:amd64: 5.4.0-6ubuntu1~16.04.2
libncurses5:amd64: 6.0+20160213-1ubuntu1
libncursesw5:amd64: 6.0+20160213-1ubuntu1
libnetfilter-conntrack3:amd64: 1.0.5-1
libnettle6:amd64: 3.2-1
libnewt0.52:amd64: 0.52.18-1ubuntu2
libnfnetlink0:amd64: 1.0.1-3
libnih-dbus1:amd64: 1.0.3-4.3ubuntu1
libnih1:amd64: 1.0.3-4.3ubuntu1
libnl-3-200:amd64: 3.2.27-1
libnl-genl-3-200:amd64: 3.2.27-1
libnspr4:amd64: 2:4.12-0ubuntu0.16.04.1
libnss3:amd64: 2:3.23-0ubuntu0.16.04.1
libnss3-nssdb: 2:3.23-0ubuntu0.16.04.1
libnuma-dev:amd64: 2.0.11-1ubuntu1
libnuma1:amd64: 2.0.11-1ubuntu1
libogg0:amd64: 1.3.2-1
libopus0:amd64: 1.1.2-1ubuntu1
libp11-kit0:amd64: 0.23.2-3
libpam-cgfs: 2.0.6-0ubuntu1~16.04.1
libpam-modules:amd64: 1.1.8-3.2ubuntu2
libpam-modules-bin: 1.1.8-3.2ubuntu2
libpam-runtime: 1.1.8-3.2ubuntu2
libpam0g:amd64: 1.1.8-3.2ubuntu2
libpango-1.0-0:amd64: 1.38.1-1
libpangocairo-1.0-0:amd64: 1.38.1-1
libpangoft2-1.0-0:amd64: 1.38.1-1
libpcap-dev: 1.7.4-2
```

(continues on next page)

(continued from previous page)

```
libpcap0.8:amd64: 1.7.4-2
libpcap0.8-dev: 1.7.4-2
libpci3:amd64: 1:3.3.1-1.1ubuntu1
libpciaccess0:amd64: 0.13.4-1
libpcre16-3:amd64: 2:8.38-3.1
libpcre3:amd64: 2:8.38-3.1
libpcre3-dev:amd64: 2:8.38-3.1
libpcre32-3:amd64: 2:8.38-3.1
libpcrecpp0v5:amd64: 2:8.38-3.1
libpcsclite1:amd64: 1.8.14-1ubuntu1.16.04.1
libperl5.22:amd64: 5.22.1-9
libpixman-1-0:amd64: 0.33.6-1
libplymouth4:amd64: 0.9.2-3ubuntu13.1
libpng12-0:amd64: 1.2.54-1ubuntu1
libpolkit-gobject-1-0:amd64: 0.105-14.1
libpopt0:amd64: 1.16-10
libprocps4:amd64: 2:3.3.10-4ubuntu2
libpthread-stubs0-dev:amd64: 0.3-4
libpulse0:amd64: 1:8.0-0ubuntu3
libpython-all-dev:amd64: 2.7.11-1
libpython-dev:amd64: 2.7.11-1
libpython-stdlib:amd64: 2.7.11-1
libpython2.7:amd64: 2.7.12-1~16.04
libpython2.7-dev:amd64: 2.7.12-1~16.04
libpython2.7-minimal:amd64: 2.7.12-1~16.04
libpython2.7-stdlib:amd64: 2.7.12-1~16.04
libpython3-stdlib:amd64: 3.5.1-3
libpython3.5:amd64: 3.5.2-2ubuntu0~16.04.1
libpython3.5-minimal:amd64: 3.5.2-2ubuntu0~16.04.1
libpython3.5-stdlib:amd64: 3.5.2-2ubuntu0~16.04.1
libquadmath0:amd64: 5.4.0-6ubuntu1~16.04.2
librados2: 10.2.2-0ubuntu0.16.04.2
librbd1: 10.2.2-0ubuntu0.16.04.2
libreadline6:amd64: 6.3-8ubuntu2
libroken18-heimdal:amd64: 1.7~git20150920+dfsg-4ubuntu1
librtmp1:amd64: 2.4+20151223.gitfa8646d-1build1
libsasl2-2:amd64: 2.1.26.dfsg1-14build1
libsasl2-modules:amd64: 2.1.26.dfsg1-14build1
libsasl2-modules-db:amd64: 2.1.26.dfsg1-14build1
libsdl1.2debian:amd64: 1.2.15+dfsg1-3
libseccomp2:amd64: 2.2.3-3ubuntu3
libselinux1:amd64: 2.4-3build2
libsemanage-common: 2.3-1build3
libsemanage1:amd64: 2.3-1build3
libsepol1:amd64: 2.4-2
libsigsegv2:amd64: 2.10-4
libslang2:amd64: 2.3.0-2ubuntu1
libsm-dev:amd64: 2:1.2.2-1
libsm6:amd64: 2:1.2.2-1
libsmartcols1:amd64: 2.27.1-6ubuntu3.1
libsndfile1:amd64: 1.0.25-10
libspice-server1:amd64: 0.12.6-4ubuntu0.1
libsqlite3-0:amd64: 3.11.0-1ubuntu1
libss2:amd64: 1.42.13-1ubuntu1
libssl1.0.0:amd64: 1.0.2g-1ubuntu4.5
libstdc++-5-dev:amd64: 5.4.0-6ubuntu1~16.04.2
libstdc++6:amd64: 5.4.0-6ubuntu1~16.04.2
libsystemd0:amd64: 229-4ubuntu10
libtasn1-6:amd64: 4.7-3ubuntu0.16.04.1
libtcl8.6:amd64: 8.6.5+dfsg-2
libtext-charwidth-perl: 0.04-7build5
```

(continues on next page)

(continued from previous page)

```
libtext-iconv-perl: 1.7-5build4
libtext-wrapi18n-perl: 0.06-7.1
libthai-data: 0.1.24-2
libthai0:amd64: 0.1.24-2
libtiff5:amd64: 4.0.6-1ubuntu0.2
libtinfo5:amd64: 6.0+20160213-1ubuntu1
libtk8.6:amd64: 8.6.5-1
libtool: 2.4.6-0.1
libtsan0:amd64: 5.4.0-6ubuntu1~16.04.2
libtxc-dxtn-s2tc0:amd64: 0~git20131104-1.1
libubsan0:amd64: 5.4.0-6ubuntu1~16.04.2
libudev1:amd64: 229-4ubuntu10
libusb-0.1-4:amd64: 2:0.1.12-28
libusb-1.0-0:amd64: 2:1.0.20-1
libusbredirparser1:amd64: 0.7.1-1
libustr-1.0-1:amd64: 1.0.4-5
libutempter0:amd64: 1.1.6-3
libuuid1:amd64: 2.27.1-6ubuntu3.1
libvorbis0a:amd64: 1.3.5-3
libvorbisenc2:amd64: 1.3.5-3
libwind0-heimdal:amd64: 1.7~git20150920+dfsg-4ubuntu1
libwrap0:amd64: 7.6.q-25
libx11-6:amd64: 2:1.6.3-1ubuntu2
libx11-data: 2:1.6.3-1ubuntu2
libx11-dev:amd64: 2:1.6.3-1ubuntu2
libx11-doc: 2:1.6.3-1ubuntu2
libx11-xcb1:amd64: 2:1.6.3-1ubuntu2
libxau-dev:amd64: 1:1.0.8-1
libxau6:amd64: 1:1.0.8-1
libxaw7:amd64: 2:1.0.13-1
libxcb-dri2-0:amd64: 1.11.1-1ubuntu1
libxcb-dri3-0:amd64: 1.11.1-1ubuntu1
libxcb-glx0:amd64: 1.11.1-1ubuntu1
libxcb-present0:amd64: 1.11.1-1ubuntu1
libxcb-render0:amd64: 1.11.1-1ubuntu1
libxcb-shape0:amd64: 1.11.1-1ubuntu1
libxcb-shm0:amd64: 1.11.1-1ubuntu1
libxcb-sync1:amd64: 1.11.1-1ubuntu1
libxcb1:amd64: 1.11.1-1ubuntu1
libxcb1-dev:amd64: 1.11.1-1ubuntu1
libxcomposite1:amd64: 1:0.4.4-1
libxcursor1:amd64: 1:1.1.14-1
libxdamage1:amd64: 1:1.1.4-2
libxdmcp-dev:amd64: 1:1.1.2-1.1
libxdmcp6:amd64: 1:1.1.2-1.1
libxen-4.6:amd64: 4.6.0-1ubuntu4.1
libxenstore3.0:amd64: 4.6.0-1ubuntu4.1
libxext6:amd64: 2:1.3.3-1
libxfixes3:amd64: 1:5.0.1-2
libxft2:amd64: 2.3.2-1
libxi6:amd64: 2:1.7.6-1
libxinerama1:amd64: 2:1.1.3-1
libxml2:amd64: 2.9.3+dfsg1-1ubuntu0.1
libxmu6:amd64: 2:1.1.2-2
libxmuu1:amd64: 2:1.1.2-2
libxpm4:amd64: 1:3.5.11-1
libxrandr2:amd64: 2:1.5.0-1
libxrender1:amd64: 1:0.9.9-0ubuntu1
libxshmfence1:amd64: 1.2-1
libxss1:amd64: 1:1.2.2-1
libxt-dev:amd64: 1:1.1.5-0ubuntu1
```

(continues on next page)

(continued from previous page)

```
libxt6:amd64: 1:1.1.5-0ubuntu1
libxtables11:amd64: 1.6.0-2ubuntu3
libxtst6:amd64: 2:1.2.2-1
libxv1:amd64: 2:1.0.10-1
libxxf86dga1:amd64: 2:1.1.4-1
libxxf86vm1:amd64: 1:1.1.4-1
libyajl2:amd64: 2.1.0-2
linux-base: 4.0ubuntu1
linux-firmware: 1.157.2
linux-generic: 4.4.0.72.78
linux-headers-4.4.0-72: 4.4.0-72.93
linux-headers-4.4.0-72-generic: 4.4.0-72.93
linux-headers-generic: 4.4.0.72.78
linux-image-4.4.0-72-generic: 4.4.0-72.93
linux-image-extra-4.4.0-72-generic: 4.4.0-72.93
linux-image-generic: 4.4.0.72.78
linux-libc-dev:amd64: 4.4.0-72.93
locales: 2.23-0ubuntu3
login: 1:4.2-3.1ubuntu5
logrotate: 3.8.7-2ubuntu2
lsb-base: 9.20160110ubuntu0.2
lsb-release: 9.20160110ubuntu0.2
lxc: 2.0.7-0ubuntu1~16.04.2
lxc-common: 2.0.7-0ubuntu1~16.04.2
lxc-templates: 2.0.7-0ubuntu1~16.04.2
lxc1: 2.0.7-0ubuntu1~16.04.2
lxcfs: 2.0.6-0ubuntu1~16.04.1
m4: 1.4.17-5
make: 4.1-6
makedev: 2.3.1-93ubuntu1
manpages: 4.04-2
manpages-dev: 4.04-2
mawk: 1.3.3-17ubuntu2
mime-support: 3.59ubuntu1
mount: 2.27.1-6ubuntu3.1
mountall: 2.54ubuntu1
msr-tools: 1.3-2
multiarch-support: 2.23-0ubuntu3
ncurses-base: 6.0+20160213-1ubuntu1
ncurses-bin: 6.0+20160213-1ubuntu1
ncurses-term: 6.0+20160213-1ubuntu1
net-tools: 1.60-26ubuntu1
netbase: 5.3
netcat-openbsd: 1.105-7ubuntu1
openjdk-8-jdk:amd64: 8u131-b11-2ubuntu1.16.04.3
openjdk-8-jdk-headless:amd64: 8u131-b11-2ubuntu1.16.04.3
openjdk-8-jre:amd64: 8u131-b11-2ubuntu1.16.04.3
openjdk-8-jre-headless:amd64: 8u131-b11-2ubuntu1.16.04.3
openssh-client: 1:7.2p2-4ubuntu2.1
openssh-server: 1:7.2p2-4ubuntu2.1
openssh-sftp-server: 1:7.2p2-4ubuntu2.1
openssl: 1.0.2g-1ubuntu4.5
os-prober: 1.70ubuntu3
passwd: 1:4.2-3.1ubuntu5
patch: 2.7.5-1
pciutils: 1:3.3.1-1.1ubuntu1
perl: 5.22.1-9
perl-base: 5.22.1-9
perl-modules-5.22: 5.22.1-9
pkg-config: 0.29.1-0ubuntu1
plymouth: 0.9.2-3ubuntu13.1
```

(continues on next page)

(continued from previous page)

```
plymouth-theme-ubuntu-text: 0.9.2-3ubuntu13.1
procps: 2:3.3.10-4ubuntu2
python: 2.7.11-1
python-all: 2.7.11-1
python-all-dev: 2.7.11-1
python-apt: 1.1.0~beta1build1
python-apt-common: 1.1.0~beta1build1
python-dev: 2.7.11-1
python-iniparse: 0.4-2.2
python-minimal: 2.7.11-1
python-pip: 8.1.1-2ubuntu0.2
python-pip-whl: 8.1.1-2ubuntu0.2
python-pkg-resources: 20.7.0-1
python-setuptools: 20.7.0-1
python-six: 1.10.0-3
python-virtualenv: 15.0.1+ds-3
python-wheel: 0.29.0-1
python2.7: 2.7.12-1~16.04
python2.7-dev: 2.7.12-1~16.04
python2.7-minimal: 2.7.12-1~16.04
python3: 3.5.1-3
python3-apt: 1.1.0~beta1build1
python3-chardet: 2.3.0-2
python3-dbus: 1.2.0-3
python3-gi: 3.20.0-0ubuntu1
python3-lxc: 2.0.7-0ubuntu1~16.04.2
python3-minimal: 3.5.1-3
python3-pkg-resources: 20.7.0-1
python3-requests: 2.9.1-3
python3-six: 1.10.0-3
python3-urllib3: 1.13.1-2ubuntu0.16.04.1
python3-virtualenv: 15.0.1+ds-3
python3.5: 3.5.2-2ubuntu0~16.04.1
python3.5-minimal: 3.5.2-2ubuntu0~16.04.1
qemu-block-extra:amd64: 1:2.5+dfsg-5ubuntu10.5
qemu-system-common: 1:2.5+dfsg-5ubuntu10.5
qemu-system-x86: 1:2.5+dfsg-5ubuntu10.5
qemu-utils: 1:2.5+dfsg-5ubuntu10.5
readline-common: 6.3-8ubuntu2
rename: 0.20-4
resolvconf: 1.78ubuntu2
rsync: 3.1.1-3ubuntu1
rsyslog: 8.16.0-1ubuntu3
screen: 4.3.1-2build1
seabios: 1.8.2-1ubuntu1
sed: 4.2.2-7
sensible-utils: 0.0.9
sgml-base: 1.26+nmu4ubuntu1
shared-mime-info: 1.5-2ubuntu0.1
sharutils: 1:4.15.2-1
socat: 1.7.3.1-1
ssh-import-id: 5.5-0ubuntu1
sudo: 1.8.16-0ubuntu1.1
systemd: 229-4ubuntu10
systemd-sysv: 229-4ubuntu10
sysv-rc: 2.88dsf-59.3ubuntu2
sysvinit-utils: 2.88dsf-59.3ubuntu2
tar: 1.28-2.1
tasksel: 3.34ubuntu3
tasksel-data: 3.34ubuntu3
tcl-expect:amd64: 5.45-7
```

(continues on next page)

(continued from previous page)

```
tcl8.6: 8.6.5+dfsg-2
tcpd: 7.6.q-25
telnet: 0.17-40
tk8.6: 8.6.5-1
tzdata: 2016f-0ubuntu0.16.04
ubuntu-keyring: 2012.05.19
ubuntu-minimal: 1.361
ucf: 3.0036
udev: 229-4ubuntu10
uidmap: 1:4.2-3.1ubuntu5.3
ureadahead: 0.100.0-19
usbutils: 1:007-4
util-linux: 2.27.1-6ubuntu3.1
uuid-runtime: 2.27.1-6ubuntu3.2
vim-common: 2:7.4.1689-3ubuntu1.1
vim-tiny: 2:7.4.1689-3ubuntu1.1
virtualenv: 15.0.1+ds-3
wamerican: 7.1-1
wget: 1.17.1-1ubuntu1.1
whiptail: 0.52.18-1ubuntu2
wireless-regdb: 2015.07.20-1ubuntu1
x11-common: 1:7.7+13ubuntu3
x11-utils: 7.7+3
x11proto-core-dev: 7.0.31-1~ubuntu16.04.1
x11proto-input-dev: 2.3.1-1
x11proto-kb-dev: 1.0.7-0ubuntu1
xauth: 1:1.0.9-1ubuntu2
xbitmaps: 1.1.1-2
xdg-user-dirs: 0.15-2ubuntu6
xkb-data: 2.16-1ubuntu1
xml-core: 0.13+nmu2
xorg-sgml-doctools: 1:1.11-1
xterm: 322-1ubuntu1
xtrans-dev: 1.3.5-1
xz-utils: 5.1.1alpha+20120614-2ubuntu2
zlib1g:amd64: 1:1.2.8.dfsg-2ubuntu4
zlib1g-dev:amd64: 1:1.2.8.dfsg-2ubuntu4
```

Kernel module listing

```
$ lsmod | sort
8250_fintek            16384  0
ablk_helper            16384  1 aesni_intel
acpi_pad               24576  0
acpi_power_meter       20480  0
aesni_intel           167936  0
aes_x86_64             20480  1 aesni_intel
ahci                   36864  0
authenc                16384  1 intel_qat
autofs4                40960  2
bridge                126976  0
coretemp               16384  0
crc32_pclmul           16384  0
crct10dif_pclmul       16384  0
cryptd                 20480  3 ghash_clmulni_intel,aesni_intel,ablk_helper
dca                    16384  2 igb,ixgbe
edac_core              53248  1 sb_edac
enclosure              16384  1 ses
enic                   81920  0
fjes                   28672  0
fnic                  106496  0
```

(continues on next page)

(continued from previous page)

```
gf128mul               16384  1 lrw
ghash_clmulni_intel    16384  0
glue_helper            16384  1 aesni_intel
hid                   118784  2 hid_generic,usbhid
hid_generic            16384  0
i2c_algo_bit           16384  1 igb
i40e                  286720  0
igb                   196608  0
igb_uio                16384  0
input_leds             16384  0
intel_powerclamp       16384  0
intel_qat             110592  2 qat_dh895xccvf,qat_dh895xcc
intel_rapl             20480  0
ip6_udp_tunnel         16384  1 vxlan
ipmi_msghandler        49152  2 ipmi_ssif,ipmi_si
ipmi_si                57344  0
ipmi_ssif              24576  0
iptable_filter         16384  1
iptable_mangle         16384  1
iptable_nat            16384  1
ip_tables              24576  3 iptable_filter,iptable_mangle,iptable_nat
ipt_MASQUERADE         16384  1
irqbypass              16384  1 kvm
ixgbe                 290816  0
joydev                 20480  0
kvm                   544768  1 kvm_intel
kvm_intel             172032  0
libahci                32768  1 ahci
libfc                 114688  2 fnic,libfcoe
libfcoe                65536  1 fnic
llc                    16384  2 stp,bridge
lpc_ich                24576  0
lrw                    16384  1 aesni_intel
mac_hid                16384  0
mdio                   16384  1 ixgbe
megaraid_sas          135168  3
mei                    98304  1 mei_me
mei_me                 36864  0
Module                  Size  Used by
nf_conntrack          106496  4 nf_nat,nf_nat_ipv4,nf_nat_masquerade_ipv4,nf_conntrack_ipv4
nf_conntrack_ipv4      16384  1
nf_defrag_ipv4         16384  1 nf_conntrack_ipv4
nf_nat                 24576  2 nf_nat_ipv4,nf_nat_masquerade_ipv4
nf_nat_ipv4            16384  1 iptable_nat
nf_nat_masquerade_ipv4    16384  1 ipt_MASQUERADE
pps_core               20480  1 ptp
ptp                    20480  3 igb,i40e,ixgbe
qat_dh895xcc           20480  0
qat_dh895xccvf         20480  0
sb_edac                32768  0
scsi_transport_fc      61440  2 fnic,libfc
ses                    20480  0
shpchp                 36864  0
stp                    16384  1 bridge
udp_tunnel             16384  1 vxlan
uio                    20480  2 uio_pci_generic,igb_uio
uio_pci_generic        16384  0
usbhid                 49152  0
veth                   16384  0
vxlan                  49152  2 i40e,ixgbe
wmi                    20480  0
```

(continues on next page)

(continued from previous page)

```
x86_pkg_temp_thermal    16384  0
x_tables               36864  6 xt_CHECKSUM,ip_tables,xt_tcpudp,ipt_MASQUERADE,iptable_filter,
↪iptable_mangle
xt_CHECKSUM            16384  1
xt_tcpudp              16384  5
```

Sysctl listing

```
$ sysctl -a
abi.vsyscall32 = 1
debug.exception-trace = 1
debug.kprobes-optimization = 1
dev.cdrom.autoclose = 1
dev.cdrom.autoeject = 0
dev.cdrom.check_media = 0
dev.cdrom.debug = 0
dev.cdrom.info = CD-ROM information, Id: cdrom.c 3.20 2003/12/17
dev.cdrom.info =
dev.cdrom.info = drive name:
dev.cdrom.info = drive speed:
dev.cdrom.info = drive # of slots:
dev.cdrom.info = Can close tray:
dev.cdrom.info = Can open tray:
dev.cdrom.info = Can lock tray:
dev.cdrom.info = Can change speed:
dev.cdrom.info = Can select disk:
dev.cdrom.info = Can read multisession:
dev.cdrom.info = Can read MCN:
dev.cdrom.info = Reports media changed:
dev.cdrom.info = Can play audio:
dev.cdrom.info = Can write CD-R:
dev.cdrom.info = Can write CD-RW:
dev.cdrom.info = Can read DVD:
dev.cdrom.info = Can write DVD-R:
dev.cdrom.info = Can write DVD-RAM:
dev.cdrom.info = Can read MRW:
dev.cdrom.info = Can write MRW:
dev.cdrom.info = Can write RAM:
dev.cdrom.info =
dev.cdrom.info =
dev.cdrom.lock = 0
dev.hpet.max-user-freq = 64
dev.mac_hid.mouse_button2_keycode = 97
dev.mac_hid.mouse_button3_keycode = 100
dev.mac_hid.mouse_button_emulation = 0
dev.raid.speed_limit_max = 200000
dev.raid.speed_limit_min = 1000
dev.scsi.logging_level = 0
fs.aio-max-nr = 65536
fs.aio-nr = 0
fs.binfmt_misc.status = enabled
fs.dentry-state = 69970     58326   45      0       0       0
fs.dir-notify-enable = 1
fs.epoll.max_user_watches = 108185784
fs.file-max = 52706330
fs.file-nr = 1224   0       52706330
fs.inode-nr = 42965 369
fs.inode-state = 42965      369     0       0       0       0       0
fs.inotify.max_queued_events = 16384
fs.inotify.max_user_instances = 128
fs.inotify.max_user_watches = 8192
```

(continues on next page)

(continued from previous page)

```
fs.lease-break-time = 45
fs.leases-enable = 1
fs.mount-max = 100000
fs.mqueue.msg_default = 10
fs.mqueue.msg_max = 10
fs.mqueue.msgsize_default = 8192
fs.mqueue.msgsize_max = 8192
fs.mqueue.queues_max = 256
fs.nr_open = 1048576
fs.overflowgid = 65534
fs.overflowuid = 65534
fs.pipe-max-size = 1048576
fs.pipe-user-pages-hard = 0
fs.pipe-user-pages-soft = 16384
fs.protected_hardlinks = 1
fs.protected_symlinks = 1
fs.quota.allocated_dquots = 0
fs.quota.cache_hits = 0
fs.quota.drops = 0
fs.quota.free_dquots = 0
fs.quota.lookups = 0
fs.quota.reads = 0
fs.quota.syncs = 0
fs.quota.writes = 0
fs.suid_dumpable = 0
kernel.acct = 4     2       30
kernel.acpi_video_flags = 0
kernel.auto_msgmni = 0
kernel.bootloader_type = 114
kernel.bootloader_version = 2
kernel.cad_pid = 1
kernel.cap_last_cap = 37
kernel.compat-log = 1
kernel.core_pattern = core
kernel.core_pipe_limit = 0
kernel.core_uses_pid = 0
kernel.ctrl-alt-del = 0
kernel.dmesg_restrict = 0
kernel.domainname = (none)
kernel.ftrace_dump_on_oops = 0
kernel.ftrace_enabled = 1
kernel.hardlockup_all_cpu_backtrace = 0
kernel.hardlockup_panic = 0
kernel.hostname = t2-sut1
kernel.hotplug =
kernel.hung_task_check_count = 4194304
kernel.hung_task_panic = 0
kernel.hung_task_timeout_secs = 120
kernel.hung_task_warnings = 10
kernel.io_delay_type = 1
kernel.kexec_load_disabled = 0
kernel.keys.gc_delay = 300
kernel.keys.maxbytes = 20000
kernel.keys.maxkeys = 200
kernel.keys.persistent_keyring_expiry = 259200
kernel.keys.root_maxbytes = 25000000
kernel.keys.root_maxkeys = 1000000
kernel.kptr_restrict = 1
kernel.kstack_depth_to_print = 12
kernel.max_lock_depth = 1024
kernel.modprobe = /sbin/modprobe
```

(continues on next page)

(continued from previous page)

```
kernel.modules_disabled = 0
kernel.moksbstate_disabled = 0
kernel.msg_next_id = -1
kernel.msgmax = 8192
kernel.msgmnb = 16384
kernel.msgmni = 32000
kernel.ngroups_max = 65536
kernel.nmi_watchdog = 1
kernel.ns_last_pid = 11764
kernel.numa_balancing = 1
kernel.numa_balancing_scan_delay_ms = 1000
kernel.numa_balancing_scan_period_max_ms = 60000
kernel.numa_balancing_scan_period_min_ms = 1000
kernel.numa_balancing_scan_size_mb = 256
kernel.osrelease = 4.4.0-72-generic
kernel.ostype = Linux
kernel.overflowgid = 65534
kernel.overflowuid = 65534
kernel.panic = 0
kernel.panic_on_io_nmi = 0
kernel.panic_on_oops = 0
kernel.panic_on_unrecovered_nmi = 0
kernel.panic_on_warn = 0
kernel.perf_cpu_time_max_percent = 25
kernel.perf_event_max_sample_rate = 12500
kernel.perf_event_mlock_kb = 516
kernel.perf_event_paranoid = 1
kernel.pid_max = 36864
kernel.poweroff_cmd = /sbin/poweroff
kernel.print-fatal-signals = 0
kernel.printk = 4   4       1       7
kernel.printk_delay = 0
kernel.printk_ratelimit = 5
kernel.printk_ratelimit_burst = 10
kernel.pty.max = 4096
kernel.pty.nr = 1
kernel.pty.reserve = 1024
kernel.random.boot_id = f683c836-6fc6-492a-a23b-62ab21895040
kernel.random.entropy_avail = 200
kernel.random.poolsize = 4096
kernel.random.read_wakeup_threshold = 64
kernel.random.urandom_min_reseed_secs = 60
kernel.random.uuid = 144ff2ba-1bc7-4836-8fb7-6aaa0ab7e65f
kernel.random.write_wakeup_threshold = 896
kernel.randomize_va_space = 0
kernel.real-root-dev = 0
kernel.sched_autogroup_enabled = 1
kernel.sched_cfs_bandwidth_slice_us = 5000
kernel.sched_child_runs_first = 0
kernel.sched_domain.cpu0.domain0.busy_factor = 32
kernel.sched_domain.cpu0.domain0.busy_idx = 3
kernel.sched_domain.cpu0.domain0.cache_nice_tries = 2
kernel.sched_domain.cpu0.domain0.flags = 25647
kernel.sched_domain.cpu0.domain0.forkexec_idx = 0
kernel.sched_domain.cpu0.domain0.idle_idx = 2
kernel.sched_domain.cpu0.domain0.imbalance_pct = 125
kernel.sched_domain.cpu0.domain0.max_interval = 72
kernel.sched_domain.cpu0.domain0.max_newidle_lb_cost = 1309
kernel.sched_domain.cpu0.domain0.min_interval = 36
kernel.sched_domain.cpu0.domain0.name = NUMA
kernel.sched_domain.cpu0.domain0.newidle_idx = 0
```

(continues on next page)

(continued from previous page)

```
kernel.sched_domain.cpu0.domain0.wake_idx = 0
kernel.sched_domain.cpu18.domain0.busy_factor = 32
kernel.sched_domain.cpu18.domain0.busy_idx = 3
kernel.sched_domain.cpu18.domain0.cache_nice_tries = 2
kernel.sched_domain.cpu18.domain0.flags = 25647
kernel.sched_domain.cpu18.domain0.forkexec_idx = 0
kernel.sched_domain.cpu18.domain0.idle_idx = 2
kernel.sched_domain.cpu18.domain0.imbalance_pct = 125
kernel.sched_domain.cpu18.domain0.max_interval = 72
kernel.sched_domain.cpu18.domain0.max_newidle_lb_cost = 2026
kernel.sched_domain.cpu18.domain0.min_interval = 36
kernel.sched_domain.cpu18.domain0.name = NUMA
kernel.sched_domain.cpu18.domain0.newidle_idx = 0
kernel.sched_domain.cpu18.domain0.wake_idx = 0
kernel.sched_latency_ns = 24000000
kernel.sched_migration_cost_ns = 500000
kernel.sched_min_granularity_ns = 3000000
kernel.sched_nr_migrate = 32
kernel.sched_rr_timeslice_ms = 25
kernel.sched_rt_period_us = 1000000
kernel.sched_rt_runtime_us = 950000
kernel.sched_shares_window_ns = 10000000
kernel.sched_time_avg_ms = 1000
kernel.sched_tunable_scaling = 1
kernel.sched_wakeup_granularity_ns = 4000000
kernel.secure_boot = 0
kernel.sem = 32000  1024000000      500     32000
kernel.sem_next_id = -1
kernel.sg-big-buff = 32768
kernel.shm_next_id = -1
kernel.shm_rmid_forced = 0
kernel.shmall = 18446744073692774399
kernel.shmmax = 8589934592
kernel.shmmni = 4096
kernel.soft_watchdog = 1
kernel.softlockup_all_cpu_backtrace = 0
kernel.softlockup_panic = 0
kernel.stack_tracer_enabled = 0
kernel.sysctl_writes_strict = 0
kernel.sysrq = 176
kernel.tainted = 12288
kernel.threads-max = 4126960
kernel.timer_migration = 1
kernel.traceoff_on_warning = 0
kernel.tracepoint_printk = 0
kernel.unknown_nmi_panic = 0
kernel.unprivileged_bpf_disabled = 0
kernel.unprivileged_userns_apparmor_policy = 1
kernel.unprivileged_userns_clone = 1
kernel.usermodehelper.bset = 4294967295     63
kernel.usermodehelper.inheritable = 4294967295      63
kernel.version = #93-Ubuntu SMP Fri Mar 31 14:07:41 UTC 2017
kernel.watchdog = 1
kernel.watchdog_cpumask = 0,18
kernel.watchdog_thresh = 10
kernel.yama.ptrace_scope = 1
net.core.bpf_jit_enable = 0
net.core.busy_poll = 0
net.core.busy_read = 0
net.core.default_qdisc = pfifo_fast
net.core.dev_weight = 64
```

(continues on next page)

(continued from previous page)

```
net.core.flow_limit_cpu_bitmap = 0,00000000
net.core.flow_limit_table_len = 4096
net.core.max_skb_frags = 17
net.core.message_burst = 10
net.core.message_cost = 5
net.core.netdev_budget = 300
net.core.netdev_max_backlog = 1000
net.core.netdev_rss_key = 
↪29:61:61:e6:4e:d5:d0:a2:dc:81:6a:c8:44:1b:e2:8d:c8:6f:6a:2b:64:62:98:08:bb:63:48:8e:96:d1:6a:15:32:ca:da:8d:3c:0a:e
net.core.netdev_tstamp_prequeue = 1
net.core.optmem_max = 20480
net.core.rmem_default = 212992
net.core.rmem_max = 212992
net.core.rps_sock_flow_entries = 0
net.core.somaxconn = 128
net.core.tstamp_allow_data = 1
net.core.warnings = 0
net.core.wmem_default = 212992
net.core.wmem_max = 212992
net.core.xfrm_acq_expires = 30
net.core.xfrm_aevent_etime = 10
net.core.xfrm_aevent_rseqth = 2
net.core.xfrm_larval_drop = 1
net.fan.vxlan = 4
net.ipv4.cipso_cache_bucket_size = 10
net.ipv4.cipso_cache_enable = 1
net.ipv4.cipso_rbm_optfmt = 0
net.ipv4.cipso_rbm_strictvalid = 1
net.ipv4.conf.all.accept_local = 0
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.all.arp_accept = 0
net.ipv4.conf.all.arp_announce = 0
net.ipv4.conf.all.arp_filter = 0
net.ipv4.conf.all.arp_ignore = 0
net.ipv4.conf.all.arp_notify = 0
net.ipv4.conf.all.bootp_relay = 0
net.ipv4.conf.all.disable_policy = 0
net.ipv4.conf.all.disable_xfrm = 0
net.ipv4.conf.all.force_igmp_version = 0
net.ipv4.conf.all.forwarding = 1
net.ipv4.conf.all.igmpv2_unsolicited_report_interval = 10000
net.ipv4.conf.all.igmpv3_unsolicited_report_interval = 1000
net.ipv4.conf.all.ignore_routes_with_linkdown = 0
net.ipv4.conf.all.log_martians = 0
net.ipv4.conf.all.mc_forwarding = 0
net.ipv4.conf.all.medium_id = 0
net.ipv4.conf.all.promote_secondaries = 0
net.ipv4.conf.all.proxy_arp = 0
net.ipv4.conf.all.proxy_arp_pvlan = 0
net.ipv4.conf.all.route_localnet = 0
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.all.secure_redirects = 1
net.ipv4.conf.all.send_redirects = 1
net.ipv4.conf.all.shared_media = 1
net.ipv4.conf.all.src_valid_mark = 0
net.ipv4.conf.all.tag = 0
net.ipv4.conf.default.accept_local = 0
net.ipv4.conf.default.accept_redirects = 1
net.ipv4.conf.default.accept_source_route = 1
net.ipv4.conf.default.arp_accept = 0
```

(continues on next page)

(continued from previous page)

```
net.ipv4.conf.default.arp_announce = 0
net.ipv4.conf.default.arp_filter = 0
net.ipv4.conf.default.arp_ignore = 0
net.ipv4.conf.default.arp_notify = 0
net.ipv4.conf.default.bootp_relay = 0
net.ipv4.conf.default.disable_policy = 0
net.ipv4.conf.default.disable_xfrm = 0
net.ipv4.conf.default.force_igmp_version = 0
net.ipv4.conf.default.forwarding = 1
net.ipv4.conf.default.igmpv2_unsolicited_report_interval = 10000
net.ipv4.conf.default.igmpv3_unsolicited_report_interval = 1000
net.ipv4.conf.default.ignore_routes_with_linkdown = 0
net.ipv4.conf.default.log_martians = 0
net.ipv4.conf.default.mc_forwarding = 0
net.ipv4.conf.default.medium_id = 0
net.ipv4.conf.default.promote_secondaries = 0
net.ipv4.conf.default.proxy_arp = 0
net.ipv4.conf.default.proxy_arp_pvlan = 0
net.ipv4.conf.default.route_localnet = 0
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.default.secure_redirects = 1
net.ipv4.conf.default.send_redirects = 1
net.ipv4.conf.default.shared_media = 1
net.ipv4.conf.default.src_valid_mark = 0
net.ipv4.conf.default.tag = 0
net.ipv4.conf.enp25s0f0.accept_local = 0
net.ipv4.conf.enp25s0f0.accept_redirects = 1
net.ipv4.conf.enp25s0f0.accept_source_route = 1
net.ipv4.conf.enp25s0f0.arp_accept = 0
net.ipv4.conf.enp25s0f0.arp_announce = 0
net.ipv4.conf.enp25s0f0.arp_filter = 0
net.ipv4.conf.enp25s0f0.arp_ignore = 0
net.ipv4.conf.enp25s0f0.arp_notify = 0
net.ipv4.conf.enp25s0f0.bootp_relay = 0
net.ipv4.conf.enp25s0f0.disable_policy = 0
net.ipv4.conf.enp25s0f0.disable_xfrm = 0
net.ipv4.conf.enp25s0f0.force_igmp_version = 0
net.ipv4.conf.enp25s0f0.forwarding = 1
net.ipv4.conf.enp25s0f0.igmpv2_unsolicited_report_interval = 10000
net.ipv4.conf.enp25s0f0.igmpv3_unsolicited_report_interval = 1000
net.ipv4.conf.enp25s0f0.ignore_routes_with_linkdown = 0
net.ipv4.conf.enp25s0f0.log_martians = 0
net.ipv4.conf.enp25s0f0.mc_forwarding = 0
net.ipv4.conf.enp25s0f0.medium_id = 0
net.ipv4.conf.enp25s0f0.promote_secondaries = 0
net.ipv4.conf.enp25s0f0.proxy_arp = 0
net.ipv4.conf.enp25s0f0.proxy_arp_pvlan = 0
net.ipv4.conf.enp25s0f0.route_localnet = 0
net.ipv4.conf.enp25s0f0.rp_filter = 1
net.ipv4.conf.enp25s0f0.secure_redirects = 1
net.ipv4.conf.enp25s0f0.send_redirects = 1
net.ipv4.conf.enp25s0f0.shared_media = 1
net.ipv4.conf.enp25s0f0.src_valid_mark = 0
net.ipv4.conf.enp25s0f0.tag = 0
net.ipv4.conf.lo.accept_local = 0
net.ipv4.conf.lo.accept_redirects = 1
net.ipv4.conf.lo.accept_source_route = 1
net.ipv4.conf.lo.arp_accept = 0
net.ipv4.conf.lo.arp_announce = 0
net.ipv4.conf.lo.arp_filter = 0
net.ipv4.conf.lo.arp_ignore = 0
```

(continues on next page)

(continued from previous page)

```
net.ipv4.conf.lo.arp_notify = 0
net.ipv4.conf.lo.bootp_relay = 0
net.ipv4.conf.lo.disable_policy = 1
net.ipv4.conf.lo.disable_xfrm = 1
net.ipv4.conf.lo.force_igmp_version = 0
net.ipv4.conf.lo.forwarding = 1
net.ipv4.conf.lo.igmpv2_unsolicited_report_interval = 10000
net.ipv4.conf.lo.igmpv3_unsolicited_report_interval = 1000
net.ipv4.conf.lo.ignore_routes_with_linkdown = 0
net.ipv4.conf.lo.log_martians = 0
net.ipv4.conf.lo.mc_forwarding = 0
net.ipv4.conf.lo.medium_id = 0
net.ipv4.conf.lo.promote_secondaries = 0
net.ipv4.conf.lo.proxy_arp = 0
net.ipv4.conf.lo.proxy_arp_pvlan = 0
net.ipv4.conf.lo.route_localnet = 0
net.ipv4.conf.lo.rp_filter = 0
net.ipv4.conf.lo.secure_redirects = 1
net.ipv4.conf.lo.send_redirects = 1
net.ipv4.conf.lo.shared_media = 1
net.ipv4.conf.lo.src_valid_mark = 0
net.ipv4.conf.lo.tag = 0
net.ipv4.conf.lxcbr0.accept_local = 0
net.ipv4.conf.lxcbr0.accept_redirects = 1
net.ipv4.conf.lxcbr0.accept_source_route = 1
net.ipv4.conf.lxcbr0.arp_accept = 0
net.ipv4.conf.lxcbr0.arp_announce = 0
net.ipv4.conf.lxcbr0.arp_filter = 0
net.ipv4.conf.lxcbr0.arp_ignore = 0
net.ipv4.conf.lxcbr0.arp_notify = 0
net.ipv4.conf.lxcbr0.bootp_relay = 0
net.ipv4.conf.lxcbr0.disable_policy = 0
net.ipv4.conf.lxcbr0.disable_xfrm = 0
net.ipv4.conf.lxcbr0.force_igmp_version = 0
net.ipv4.conf.lxcbr0.forwarding = 1
net.ipv4.conf.lxcbr0.igmpv2_unsolicited_report_interval = 10000
net.ipv4.conf.lxcbr0.igmpv3_unsolicited_report_interval = 1000
net.ipv4.conf.lxcbr0.ignore_routes_with_linkdown = 0
net.ipv4.conf.lxcbr0.log_martians = 0
net.ipv4.conf.lxcbr0.mc_forwarding = 0
net.ipv4.conf.lxcbr0.medium_id = 0
net.ipv4.conf.lxcbr0.promote_secondaries = 0
net.ipv4.conf.lxcbr0.proxy_arp = 0
net.ipv4.conf.lxcbr0.proxy_arp_pvlan = 0
net.ipv4.conf.lxcbr0.route_localnet = 0
net.ipv4.conf.lxcbr0.rp_filter = 1
net.ipv4.conf.lxcbr0.secure_redirects = 1
net.ipv4.conf.lxcbr0.send_redirects = 1
net.ipv4.conf.lxcbr0.shared_media = 1
net.ipv4.conf.lxcbr0.src_valid_mark = 0
net.ipv4.conf.lxcbr0.tag = 0
net.ipv4.fwmark_reflect = 0
net.ipv4.icmp_echo_ignore_all = 0
net.ipv4.icmp_echo_ignore_broadcasts = 1
net.ipv4.icmp_errors_use_inbound_ifaddr = 0
net.ipv4.icmp_ignore_bogus_error_responses = 1
net.ipv4.icmp_msgs_burst = 50
net.ipv4.icmp_msgs_per_sec = 1000
net.ipv4.icmp_ratelimit = 1000
net.ipv4.icmp_ratemask = 6168
net.ipv4.igmp_link_local_mcast_reports = 1
```

(continues on next page)

(continued from previous page)

```
net.ipv4.igmp_max_memberships = 20
net.ipv4.igmp_max_msf = 10
net.ipv4.igmp_qrv = 2
net.ipv4.inet_peer_maxttl = 600
net.ipv4.inet_peer_minttl = 120
net.ipv4.inet_peer_threshold = 65664
net.ipv4.ip_default_ttl = 64
net.ipv4.ip_dynaddr = 0
net.ipv4.ip_early_demux = 1
net.ipv4.ip_forward = 1
net.ipv4.ip_forward_use_pmtu = 0
net.ipv4.ip_local_port_range = 32768        60999
net.ipv4.ip_local_reserved_ports =
net.ipv4.ip_no_pmtu_disc = 0
net.ipv4.ip_nonlocal_bind = 0
net.ipv4.ipfrag_high_thresh = 4194304
net.ipv4.ipfrag_low_thresh = 3145728
net.ipv4.ipfrag_max_dist = 64
net.ipv4.ipfrag_secret_interval = 0
net.ipv4.ipfrag_time = 30
net.ipv4.neigh.default.anycast_delay = 100
net.ipv4.neigh.default.app_solicit = 0
net.ipv4.neigh.default.base_reachable_time_ms = 30000
net.ipv4.neigh.default.delay_first_probe_time = 5
net.ipv4.neigh.default.gc_interval = 30
net.ipv4.neigh.default.gc_stale_time = 60
net.ipv4.neigh.default.gc_thresh1 = 128
net.ipv4.neigh.default.gc_thresh2 = 512
net.ipv4.neigh.default.gc_thresh3 = 1024
net.ipv4.neigh.default.locktime = 100
net.ipv4.neigh.default.mcast_resolicit = 0
net.ipv4.neigh.default.mcast_solicit = 3
net.ipv4.neigh.default.proxy_delay = 80
net.ipv4.neigh.default.proxy_qlen = 64
net.ipv4.neigh.default.retrans_time_ms = 1000
net.ipv4.neigh.default.ucast_solicit = 3
net.ipv4.neigh.default.unres_qlen = 31
net.ipv4.neigh.default.unres_qlen_bytes = 65536
net.ipv4.neigh.enp25s0f0.anycast_delay = 100
net.ipv4.neigh.enp25s0f0.app_solicit = 0
net.ipv4.neigh.enp25s0f0.base_reachable_time_ms = 30000
net.ipv4.neigh.enp25s0f0.delay_first_probe_time = 5
net.ipv4.neigh.enp25s0f0.gc_stale_time = 60
net.ipv4.neigh.enp25s0f0.locktime = 100
net.ipv4.neigh.enp25s0f0.mcast_resolicit = 0
net.ipv4.neigh.enp25s0f0.mcast_solicit = 3
net.ipv4.neigh.enp25s0f0.proxy_delay = 80
net.ipv4.neigh.enp25s0f0.proxy_qlen = 64
net.ipv4.neigh.enp25s0f0.retrans_time_ms = 1000
net.ipv4.neigh.enp25s0f0.ucast_solicit = 3
net.ipv4.neigh.enp25s0f0.unres_qlen = 31
net.ipv4.neigh.enp25s0f0.unres_qlen_bytes = 65536
net.ipv4.neigh.lo.anycast_delay = 100
net.ipv4.neigh.lo.app_solicit = 0
net.ipv4.neigh.lo.base_reachable_time_ms = 30000
net.ipv4.neigh.lo.delay_first_probe_time = 5
net.ipv4.neigh.lo.gc_stale_time = 60
net.ipv4.neigh.lo.locktime = 100
net.ipv4.neigh.lo.mcast_resolicit = 0
net.ipv4.neigh.lo.mcast_solicit = 3
net.ipv4.neigh.lo.proxy_delay = 80
```

(continues on next page)

(continued from previous page)

```
net.ipv4.neigh.lo.proxy_qlen = 64
net.ipv4.neigh.lo.retrans_time_ms = 1000
net.ipv4.neigh.lo.ucast_solicit = 3
net.ipv4.neigh.lo.unres_qlen = 31
net.ipv4.neigh.lo.unres_qlen_bytes = 65536
net.ipv4.neigh.lxcbr0.anycast_delay = 100
net.ipv4.neigh.lxcbr0.app_solicit = 0
net.ipv4.neigh.lxcbr0.base_reachable_time_ms = 30000
net.ipv4.neigh.lxcbr0.delay_first_probe_time = 5
net.ipv4.neigh.lxcbr0.gc_stale_time = 60
net.ipv4.neigh.lxcbr0.locktime = 100
net.ipv4.neigh.lxcbr0.mcast_resolicit = 0
net.ipv4.neigh.lxcbr0.mcast_solicit = 3
net.ipv4.neigh.lxcbr0.proxy_delay = 80
net.ipv4.neigh.lxcbr0.proxy_qlen = 64
net.ipv4.neigh.lxcbr0.retrans_time_ms = 1000
net.ipv4.neigh.lxcbr0.ucast_solicit = 3
net.ipv4.neigh.lxcbr0.unres_qlen = 31
net.ipv4.neigh.lxcbr0.unres_qlen_bytes = 65536
net.ipv4.ping_group_range = 1       0
net.ipv4.route.error_burst = 1250
net.ipv4.route.error_cost = 250
net.ipv4.route.gc_elasticity = 8
net.ipv4.route.gc_interval = 60
net.ipv4.route.gc_min_interval = 0
net.ipv4.route.gc_min_interval_ms = 500
net.ipv4.route.gc_thresh = -1
net.ipv4.route.gc_timeout = 300
net.ipv4.route.max_size = 2147483647
net.ipv4.route.min_adv_mss = 256
net.ipv4.route.min_pmtu = 552
net.ipv4.route.mtu_expires = 600
net.ipv4.route.redirect_load = 5
net.ipv4.route.redirect_number = 9
net.ipv4.route.redirect_silence = 5120
net.ipv4.tcp_abort_on_overflow = 0
net.ipv4.tcp_adv_win_scale = 1
net.ipv4.tcp_allowed_congestion_control = cubic reno
net.ipv4.tcp_app_win = 31
net.ipv4.tcp_autocorking = 1
net.ipv4.tcp_available_congestion_control = cubic reno
net.ipv4.tcp_base_mss = 1024
net.ipv4.tcp_challenge_ack_limit = 1000
net.ipv4.tcp_congestion_control = cubic
net.ipv4.tcp_dsack = 1
net.ipv4.tcp_early_retrans = 3
net.ipv4.tcp_ecn = 2
net.ipv4.tcp_ecn_fallback = 1
net.ipv4.tcp_fack = 1
net.ipv4.tcp_fastopen = 1
net.ipv4.tcp_fastopen_key = 00000000-00000000-00000000-00000000
net.ipv4.tcp_fin_timeout = 60
net.ipv4.tcp_frto = 2
net.ipv4.tcp_fwmark_accept = 0
net.ipv4.tcp_invalid_ratelimit = 500
net.ipv4.tcp_keepalive_intvl = 75
net.ipv4.tcp_keepalive_probes = 9
net.ipv4.tcp_keepalive_time = 7200
net.ipv4.tcp_limit_output_bytes = 262144
net.ipv4.tcp_low_latency = 0
net.ipv4.tcp_max_orphans = 262144
```

(continues on next page)

(continued from previous page)

```
net.ipv4.tcp_max_reordering = 300
net.ipv4.tcp_max_syn_backlog = 2048
net.ipv4.tcp_max_tw_buckets = 262144
net.ipv4.tcp_mem = 6188856  8251809 12377712
net.ipv4.tcp_min_rtt_wlen = 300
net.ipv4.tcp_min_tso_segs = 2
net.ipv4.tcp_moderate_rcvbuf = 1
net.ipv4.tcp_mtu_probing = 0
net.ipv4.tcp_no_metrics_save = 0
net.ipv4.tcp_notsent_lowat = -1
net.ipv4.tcp_orphan_retries = 0
net.ipv4.tcp_pacing_ca_ratio = 120
net.ipv4.tcp_pacing_ss_ratio = 200
net.ipv4.tcp_probe_interval = 600
net.ipv4.tcp_probe_threshold = 8
net.ipv4.tcp_recovery = 1
net.ipv4.tcp_reordering = 3
net.ipv4.tcp_retrans_collapse = 1
net.ipv4.tcp_retries1 = 3
net.ipv4.tcp_retries2 = 15
net.ipv4.tcp_rfc1337 = 0
net.ipv4.tcp_rmem = 4096    87380   6291456
net.ipv4.tcp_sack = 1
net.ipv4.tcp_slow_start_after_idle = 1
net.ipv4.tcp_stdurg = 0
net.ipv4.tcp_syn_retries = 6
net.ipv4.tcp_synack_retries = 5
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_thin_dupack = 0
net.ipv4.tcp_thin_linear_timeouts = 0
net.ipv4.tcp_timestamps = 1
net.ipv4.tcp_tso_win_divisor = 3
net.ipv4.tcp_tw_recycle = 0
net.ipv4.tcp_tw_reuse = 0
net.ipv4.tcp_window_scaling = 1
net.ipv4.tcp_wmem = 4096    16384   4194304
net.ipv4.tcp_workaround_signed_windows = 0
net.ipv4.udp_mem = 12377712 16503618        24755424
net.ipv4.udp_rmem_min = 4096
net.ipv4.udp_wmem_min = 4096
net.ipv4.xfrm4_gc_thresh = 2147483647
net.ipv6.anycast_src_echo_reply = 0
net.ipv6.auto_flowlabels = 1
net.ipv6.bindv6only = 0
net.ipv6.conf.all.accept_dad = 1
net.ipv6.conf.all.accept_ra = 1
net.ipv6.conf.all.accept_ra_defrtr = 1
net.ipv6.conf.all.accept_ra_from_local = 0
net.ipv6.conf.all.accept_ra_min_hop_limit = 1
net.ipv6.conf.all.accept_ra_mtu = 1
net.ipv6.conf.all.accept_ra_pinfo = 1
net.ipv6.conf.all.accept_ra_rt_info_max_plen = 0
net.ipv6.conf.all.accept_ra_rtr_pref = 1
net.ipv6.conf.all.accept_redirects = 1
net.ipv6.conf.all.accept_source_route = 0
net.ipv6.conf.all.autoconf = 1
net.ipv6.conf.all.dad_transmits = 1
net.ipv6.conf.all.disable_ipv6 = 0
net.ipv6.conf.all.force_mld_version = 0
net.ipv6.conf.all.force_tllao = 0
net.ipv6.conf.all.forwarding = 0
```

(continues on next page)

(continued from previous page)

```
net.ipv6.conf.all.hop_limit = 64
net.ipv6.conf.all.ignore_routes_with_linkdown = 0
net.ipv6.conf.all.max_addresses = 16
net.ipv6.conf.all.max_desync_factor = 600
net.ipv6.conf.all.mc_forwarding = 0
net.ipv6.conf.all.mldv1_unsolicited_report_interval = 10000
net.ipv6.conf.all.mldv2_unsolicited_report_interval = 1000
net.ipv6.conf.all.mtu = 1280
net.ipv6.conf.all.ndisc_notify = 0
net.ipv6.conf.all.proxy_ndp = 0
net.ipv6.conf.all.regen_max_retry = 3
net.ipv6.conf.all.router_probe_interval = 60
net.ipv6.conf.all.router_solicitation_delay = 1
net.ipv6.conf.all.router_solicitation_interval = 4
net.ipv6.conf.all.router_solicitations = 3
sysctl: reading key "net.ipv6.conf.all.stable_secret"
net.ipv6.conf.all.suppress_frag_ndisc = 1
net.ipv6.conf.all.temp_prefered_lft = 86400
net.ipv6.conf.all.temp_valid_lft = 604800
net.ipv6.conf.all.use_oif_addrs_only = 0
net.ipv6.conf.all.use_tempaddr = 2
net.ipv6.conf.default.accept_dad = 1
net.ipv6.conf.default.accept_ra = 1
net.ipv6.conf.default.accept_ra_defrtr = 1
net.ipv6.conf.default.accept_ra_from_local = 0
net.ipv6.conf.default.accept_ra_min_hop_limit = 1
net.ipv6.conf.default.accept_ra_mtu = 1
net.ipv6.conf.default.accept_ra_pinfo = 1
net.ipv6.conf.default.accept_ra_rt_info_max_plen = 0
net.ipv6.conf.default.accept_ra_rtr_pref = 1
net.ipv6.conf.default.accept_redirects = 1
net.ipv6.conf.default.accept_source_route = 0
net.ipv6.conf.default.autoconf = 1
net.ipv6.conf.default.dad_transmits = 1
net.ipv6.conf.default.disable_ipv6 = 0
net.ipv6.conf.default.force_mld_version = 0
net.ipv6.conf.default.force_tllao = 0
net.ipv6.conf.default.forwarding = 0
net.ipv6.conf.default.hop_limit = 64
net.ipv6.conf.default.ignore_routes_with_linkdown = 0
net.ipv6.conf.default.max_addresses = 16
net.ipv6.conf.default.max_desync_factor = 600
net.ipv6.conf.default.mc_forwarding = 0
net.ipv6.conf.default.mldv1_unsolicited_report_interval = 10000
net.ipv6.conf.default.mldv2_unsolicited_report_interval = 1000
net.ipv6.conf.default.mtu = 1280
net.ipv6.conf.default.ndisc_notify = 0
net.ipv6.conf.default.proxy_ndp = 0
net.ipv6.conf.default.regen_max_retry = 3
net.ipv6.conf.default.router_probe_interval = 60
net.ipv6.conf.default.router_solicitation_delay = 1
net.ipv6.conf.default.router_solicitation_interval = 4
net.ipv6.conf.default.router_solicitations = 3
sysctl: reading key "net.ipv6.conf.default.stable_secret"
net.ipv6.conf.default.suppress_frag_ndisc = 1
net.ipv6.conf.default.temp_prefered_lft = 86400
net.ipv6.conf.default.temp_valid_lft = 604800
net.ipv6.conf.default.use_oif_addrs_only = 0
net.ipv6.conf.default.use_tempaddr = 2
net.ipv6.conf.enp25s0f0.accept_dad = 1
net.ipv6.conf.enp25s0f0.accept_ra = 1
```

(continues on next page)

(continued from previous page)

```
net.ipv6.conf.enp25s0f0.accept_ra_defrtr = 1
net.ipv6.conf.enp25s0f0.accept_ra_from_local = 0
net.ipv6.conf.enp25s0f0.accept_ra_min_hop_limit = 1
net.ipv6.conf.enp25s0f0.accept_ra_mtu = 1
net.ipv6.conf.enp25s0f0.accept_ra_pinfo = 1
net.ipv6.conf.enp25s0f0.accept_ra_rt_info_max_plen = 0
net.ipv6.conf.enp25s0f0.accept_ra_rtr_pref = 1
net.ipv6.conf.enp25s0f0.accept_redirects = 1
net.ipv6.conf.enp25s0f0.accept_source_route = 0
net.ipv6.conf.enp25s0f0.autoconf = 1
net.ipv6.conf.enp25s0f0.dad_transmits = 1
net.ipv6.conf.enp25s0f0.disable_ipv6 = 0
net.ipv6.conf.enp25s0f0.force_mld_version = 0
net.ipv6.conf.enp25s0f0.force_tllao = 0
net.ipv6.conf.enp25s0f0.forwarding = 0
net.ipv6.conf.enp25s0f0.hop_limit = 64
net.ipv6.conf.enp25s0f0.ignore_routes_with_linkdown = 0
net.ipv6.conf.enp25s0f0.max_addresses = 16
net.ipv6.conf.enp25s0f0.max_desync_factor = 600
net.ipv6.conf.enp25s0f0.mc_forwarding = 0
net.ipv6.conf.enp25s0f0.mldv1_unsolicited_report_interval = 10000
net.ipv6.conf.enp25s0f0.mldv2_unsolicited_report_interval = 1000
net.ipv6.conf.enp25s0f0.mtu = 1500
net.ipv6.conf.enp25s0f0.ndisc_notify = 0
net.ipv6.conf.enp25s0f0.proxy_ndp = 0
net.ipv6.conf.enp25s0f0.regen_max_retry = 3
net.ipv6.conf.enp25s0f0.router_probe_interval = 60
net.ipv6.conf.enp25s0f0.router_solicitation_delay = 1
net.ipv6.conf.enp25s0f0.router_solicitation_interval = 4
net.ipv6.conf.enp25s0f0.router_solicitations = 3
sysctl: reading key "net.ipv6.conf.enp25s0f0.stable_secret"
net.ipv6.conf.enp25s0f0.suppress_frag_ndisc = 1
net.ipv6.conf.enp25s0f0.temp_prefered_lft = 86400
net.ipv6.conf.enp25s0f0.temp_valid_lft = 604800
net.ipv6.conf.enp25s0f0.use_oif_addrs_only = 0
net.ipv6.conf.enp25s0f0.use_tempaddr = 0
net.ipv6.conf.lo.accept_dad = -1
net.ipv6.conf.lo.accept_ra = 1
net.ipv6.conf.lo.accept_ra_defrtr = 1
net.ipv6.conf.lo.accept_ra_from_local = 0
net.ipv6.conf.lo.accept_ra_min_hop_limit = 1
net.ipv6.conf.lo.accept_ra_mtu = 1
net.ipv6.conf.lo.accept_ra_pinfo = 1
net.ipv6.conf.lo.accept_ra_rt_info_max_plen = 0
net.ipv6.conf.lo.accept_ra_rtr_pref = 1
net.ipv6.conf.lo.accept_redirects = 1
net.ipv6.conf.lo.accept_source_route = 0
net.ipv6.conf.lo.autoconf = 1
net.ipv6.conf.lo.dad_transmits = 1
net.ipv6.conf.lo.disable_ipv6 = 0
net.ipv6.conf.lo.force_mld_version = 0
net.ipv6.conf.lo.force_tllao = 0
net.ipv6.conf.lo.forwarding = 0
net.ipv6.conf.lo.hop_limit = 64
net.ipv6.conf.lo.ignore_routes_with_linkdown = 0
net.ipv6.conf.lo.max_addresses = 16
net.ipv6.conf.lo.max_desync_factor = 600
net.ipv6.conf.lo.mc_forwarding = 0
net.ipv6.conf.lo.mldv1_unsolicited_report_interval = 10000
net.ipv6.conf.lo.mldv2_unsolicited_report_interval = 1000
net.ipv6.conf.lo.mtu = 65536
```

(continues on next page)

(continued from previous page)

```
net.ipv6.conf.lo.ndisc_notify = 0
net.ipv6.conf.lo.proxy_ndp = 0
net.ipv6.conf.lo.regen_max_retry = 3
net.ipv6.conf.lo.router_probe_interval = 60
net.ipv6.conf.lo.router_solicitation_delay = 1
net.ipv6.conf.lo.router_solicitation_interval = 4
net.ipv6.conf.lo.router_solicitations = 3
sysctl: reading key "net.ipv6.conf.lo.stable_secret"
net.ipv6.conf.lo.suppress_frag_ndisc = 1
net.ipv6.conf.lo.temp_prefered_lft = 86400
net.ipv6.conf.lo.temp_valid_lft = 604800
net.ipv6.conf.lo.use_oif_addrs_only = 0
net.ipv6.conf.lo.use_tempaddr = -1
net.ipv6.conf.lxcbr0.accept_dad = 0
net.ipv6.conf.lxcbr0.accept_ra = 1
net.ipv6.conf.lxcbr0.accept_ra_defrtr = 1
net.ipv6.conf.lxcbr0.accept_ra_from_local = 0
net.ipv6.conf.lxcbr0.accept_ra_min_hop_limit = 1
net.ipv6.conf.lxcbr0.accept_ra_mtu = 1
net.ipv6.conf.lxcbr0.accept_ra_pinfo = 1
net.ipv6.conf.lxcbr0.accept_ra_rt_info_max_plen = 0
net.ipv6.conf.lxcbr0.accept_ra_rtr_pref = 1
net.ipv6.conf.lxcbr0.accept_redirects = 1
net.ipv6.conf.lxcbr0.accept_source_route = 0
net.ipv6.conf.lxcbr0.autoconf = 1
net.ipv6.conf.lxcbr0.dad_transmits = 1
net.ipv6.conf.lxcbr0.disable_ipv6 = 0
net.ipv6.conf.lxcbr0.force_mld_version = 0
net.ipv6.conf.lxcbr0.force_tllao = 0
net.ipv6.conf.lxcbr0.forwarding = 0
net.ipv6.conf.lxcbr0.hop_limit = 64
net.ipv6.conf.lxcbr0.ignore_routes_with_linkdown = 0
net.ipv6.conf.lxcbr0.max_addresses = 16
net.ipv6.conf.lxcbr0.max_desync_factor = 600
net.ipv6.conf.lxcbr0.mc_forwarding = 0
net.ipv6.conf.lxcbr0.mldv1_unsolicited_report_interval = 10000
net.ipv6.conf.lxcbr0.mldv2_unsolicited_report_interval = 1000
net.ipv6.conf.lxcbr0.mtu = 1500
net.ipv6.conf.lxcbr0.ndisc_notify = 0
net.ipv6.conf.lxcbr0.proxy_ndp = 0
net.ipv6.conf.lxcbr0.regen_max_retry = 3
net.ipv6.conf.lxcbr0.router_probe_interval = 60
net.ipv6.conf.lxcbr0.router_solicitation_delay = 1
net.ipv6.conf.lxcbr0.router_solicitation_interval = 4
net.ipv6.conf.lxcbr0.router_solicitations = 3
sysctl: reading key "net.ipv6.conf.lxcbr0.stable_secret"
net.ipv6.conf.lxcbr0.suppress_frag_ndisc = 1
net.ipv6.conf.lxcbr0.temp_prefered_lft = 86400
net.ipv6.conf.lxcbr0.temp_valid_lft = 604800
net.ipv6.conf.lxcbr0.use_oif_addrs_only = 0
net.ipv6.conf.lxcbr0.use_tempaddr = 2
net.ipv6.flowlabel_consistency = 1
net.ipv6.flowlabel_state_ranges = 0
net.ipv6.fwmark_reflect = 0
net.ipv6.icmp.ratelimit = 1000
net.ipv6.idgen_delay = 1
net.ipv6.idgen_retries = 3
net.ipv6.ip6frag_high_thresh = 4194304
net.ipv6.ip6frag_low_thresh = 3145728
net.ipv6.ip6frag_secret_interval = 0
net.ipv6.ip6frag_time = 60
```

(continues on next page)

(continued from previous page)

```
net.ipv6.ip_nonlocal_bind = 0
net.ipv6.mld_max_msf = 64
net.ipv6.mld_qrv = 2
net.ipv6.neigh.default.anycast_delay = 100
net.ipv6.neigh.default.app_solicit = 0
net.ipv6.neigh.default.base_reachable_time_ms = 30000
net.ipv6.neigh.default.delay_first_probe_time = 5
net.ipv6.neigh.default.gc_interval = 30
net.ipv6.neigh.default.gc_stale_time = 60
net.ipv6.neigh.default.gc_thresh1 = 128
net.ipv6.neigh.default.gc_thresh2 = 512
net.ipv6.neigh.default.gc_thresh3 = 1024
net.ipv6.neigh.default.locktime = 0
net.ipv6.neigh.default.mcast_resolicit = 0
net.ipv6.neigh.default.mcast_solicit = 3
net.ipv6.neigh.default.proxy_delay = 80
net.ipv6.neigh.default.proxy_qlen = 64
net.ipv6.neigh.default.retrans_time_ms = 1000
net.ipv6.neigh.default.ucast_solicit = 3
net.ipv6.neigh.default.unres_qlen = 31
net.ipv6.neigh.default.unres_qlen_bytes = 65536
net.ipv6.neigh.enp25s0f0.anycast_delay = 100
net.ipv6.neigh.enp25s0f0.app_solicit = 0
net.ipv6.neigh.enp25s0f0.base_reachable_time_ms = 30000
net.ipv6.neigh.enp25s0f0.delay_first_probe_time = 5
net.ipv6.neigh.enp25s0f0.gc_stale_time = 60
net.ipv6.neigh.enp25s0f0.locktime = 0
net.ipv6.neigh.enp25s0f0.mcast_resolicit = 0
net.ipv6.neigh.enp25s0f0.mcast_solicit = 3
net.ipv6.neigh.enp25s0f0.proxy_delay = 80
net.ipv6.neigh.enp25s0f0.proxy_qlen = 64
net.ipv6.neigh.enp25s0f0.retrans_time_ms = 1000
net.ipv6.neigh.enp25s0f0.ucast_solicit = 3
net.ipv6.neigh.enp25s0f0.unres_qlen = 31
net.ipv6.neigh.enp25s0f0.unres_qlen_bytes = 65536
net.ipv6.neigh.lo.anycast_delay = 100
net.ipv6.neigh.lo.app_solicit = 0
net.ipv6.neigh.lo.base_reachable_time_ms = 30000
net.ipv6.neigh.lo.delay_first_probe_time = 5
net.ipv6.neigh.lo.gc_stale_time = 60
net.ipv6.neigh.lo.locktime = 0
net.ipv6.neigh.lo.mcast_resolicit = 0
net.ipv6.neigh.lo.mcast_solicit = 3
net.ipv6.neigh.lo.proxy_delay = 80
net.ipv6.neigh.lo.proxy_qlen = 64
net.ipv6.neigh.lo.retrans_time_ms = 1000
net.ipv6.neigh.lo.ucast_solicit = 3
net.ipv6.neigh.lo.unres_qlen = 31
net.ipv6.neigh.lo.unres_qlen_bytes = 65536
net.ipv6.neigh.lxcbr0.anycast_delay = 100
net.ipv6.neigh.lxcbr0.app_solicit = 0
net.ipv6.neigh.lxcbr0.base_reachable_time_ms = 30000
net.ipv6.neigh.lxcbr0.delay_first_probe_time = 5
net.ipv6.neigh.lxcbr0.gc_stale_time = 60
net.ipv6.neigh.lxcbr0.locktime = 0
net.ipv6.neigh.lxcbr0.mcast_resolicit = 0
net.ipv6.neigh.lxcbr0.mcast_solicit = 3
net.ipv6.neigh.lxcbr0.proxy_delay = 80
net.ipv6.neigh.lxcbr0.proxy_qlen = 64
net.ipv6.neigh.lxcbr0.retrans_time_ms = 1000
net.ipv6.neigh.lxcbr0.ucast_solicit = 3
```

(continues on next page)

(continued from previous page)

```
net.ipv6.neigh.lxcbr0.unres_qlen = 31
net.ipv6.neigh.lxcbr0.unres_qlen_bytes = 65536
net.ipv6.route.gc_elasticity = 9
net.ipv6.route.gc_interval = 30
net.ipv6.route.gc_min_interval = 0
net.ipv6.route.gc_min_interval_ms = 500
net.ipv6.route.gc_thresh = 1024
net.ipv6.route.gc_timeout = 60
net.ipv6.route.max_size = 4096
net.ipv6.route.min_adv_mss = 1220
net.ipv6.route.mtu_expires = 600
net.ipv6.xfrm6_gc_thresh = 2147483647
net.netfilter.nf_conntrack_acct = 0
net.netfilter.nf_conntrack_buckets = 65536
net.netfilter.nf_conntrack_checksum = 1
net.netfilter.nf_conntrack_count = 2
net.netfilter.nf_conntrack_events = 1
net.netfilter.nf_conntrack_expect_max = 1024
net.netfilter.nf_conntrack_generic_timeout = 600
net.netfilter.nf_conntrack_helper = 1
net.netfilter.nf_conntrack_icmp_timeout = 30
net.netfilter.nf_conntrack_log_invalid = 0
net.netfilter.nf_conntrack_max = 262144
net.netfilter.nf_conntrack_tcp_be_liberal = 0
net.netfilter.nf_conntrack_tcp_loose = 1
net.netfilter.nf_conntrack_tcp_max_retrans = 3
net.netfilter.nf_conntrack_tcp_timeout_close = 10
net.netfilter.nf_conntrack_tcp_timeout_close_wait = 60
net.netfilter.nf_conntrack_tcp_timeout_established = 432000
net.netfilter.nf_conntrack_tcp_timeout_fin_wait = 120
net.netfilter.nf_conntrack_tcp_timeout_last_ack = 30
net.netfilter.nf_conntrack_tcp_timeout_max_retrans = 300
net.netfilter.nf_conntrack_tcp_timeout_syn_recv = 60
net.netfilter.nf_conntrack_tcp_timeout_syn_sent = 120
net.netfilter.nf_conntrack_tcp_timeout_time_wait = 120
net.netfilter.nf_conntrack_tcp_timeout_unacknowledged = 300
net.netfilter.nf_conntrack_timestamp = 0
net.netfilter.nf_conntrack_udp_timeout = 30
net.netfilter.nf_conntrack_udp_timeout_stream = 180
net.netfilter.nf_log.0 = NONE
net.netfilter.nf_log.1 = NONE
net.netfilter.nf_log.10 = NONE
net.netfilter.nf_log.11 = NONE
net.netfilter.nf_log.12 = NONE
net.netfilter.nf_log.2 = NONE
net.netfilter.nf_log.3 = NONE
net.netfilter.nf_log.4 = NONE
net.netfilter.nf_log.5 = NONE
net.netfilter.nf_log.6 = NONE
net.netfilter.nf_log.7 = NONE
net.netfilter.nf_log.8 = NONE
net.netfilter.nf_log.9 = NONE
net.nf_conntrack_max = 262144
net.unix.max_dgram_qlen = 512
vm.admin_reserve_kbytes = 8192
vm.block_dump = 0
vm.compact_unevictable_allowed = 1
vm.dirty_background_bytes = 0
vm.dirty_background_ratio = 10
vm.dirty_bytes = 0
vm.dirty_expire_centisecs = 3000
```

(continues on next page)

(continued from previous page)

```
vm.dirty_ratio = 20
vm.dirty_writeback_centisecs = 500
vm.dirtytime_expire_seconds = 43200
vm.drop_caches = 0
vm.extfrag_threshold = 500
vm.hugepages_treat_as_movable = 0
vm.hugetlb_shm_group = 0
vm.laptop_mode = 0
vm.legacy_va_layout = 0
vm.lowmem_reserve_ratio = 256       256     32      1
vm.max_map_count = 200000
vm.memory_failure_early_kill = 0
vm.memory_failure_recovery = 1
vm.min_free_kbytes = 90112
vm.min_slab_ratio = 5
vm.min_unmapped_ratio = 1
vm.mmap_min_addr = 65536
vm.nr_hugepages = 4096
vm.nr_hugepages_mempolicy = 4096
vm.nr_overcommit_hugepages = 0
vm.nr_pdflush_threads = 0
vm.numa_zonelist_order = default
vm.oom_dump_tasks = 1
vm.oom_kill_allocating_task = 0
vm.overcommit_kbytes = 0
vm.overcommit_memory = 0
vm.overcommit_ratio = 50
vm.page-cluster = 3
vm.panic_on_oom = 0
vm.percpu_pagelist_fraction = 0
vm.stat_interval = 1
vm.swappiness = 0
vm.user_reserve_kbytes = 131072
vm.vfs_cache_pressure = 100
vm.zone_reclaim_mode = 0
```

Services listing

```
$ service --status-all
 [ + ]  apparmor
 [ - ]  bootmisc.sh
 [ - ]  checkfs.sh
 [ - ]  checkroot-bootclean.sh
 [ - ]  checkroot.sh
 [ + ]  console-setup
 [ + ]  cpufrequtils
 [ + ]  cron
 [ + ]  dbus
 [ + ]  docker
 [ + ]  ebtables
 [ + ]  grub-common
 [ - ]  hostname.sh
 [ - ]  hwclock.sh
 [ + ]  keyboard-setup
 [ - ]  killprocs
 [ + ]  kmod
 [ + ]  loadcpufreq
 [ + ]  lxcfs
 [ - ]  mountall-bootclean.sh
 [ - ]  mountall.sh
 [ - ]  mountdevsubfs.sh
```

(continues on next page)

(continued from previous page)

```
[ - ]  mountkernfs.sh
[ - ]  mountnfs-bootclean.sh
[ - ]  mountnfs.sh
[ + ]  networking
[ - ]  ondemand
[ - ]  plymouth
[ - ]  plymouth-log
[ + ]  procps
[ + ]  qemu-kvm
[ - ]  rc.local
[ + ]  resolvconf
[ - ]  rsync
[ + ]  rsyslog
[ - ]  screen-cleanup
[ - ]  sendsigs
[ + ]  ssh
[ + ]  udev
[ - ]  umountfs
[ - ]  umountnfs.sh
[ - ]  umountroot
[ + ]  urandom
[ - ]  uuidd
[ - ]  x11-common
```

Host CFS optimizations (QEMU+VPP)

Applying CFS scheduler tuning on all Qemu vcpu worker threads (those are handling testpmd - pmd threads) and VPP PMD worker threads. List of VPP PMD threads can be obtained e.g. from:

```
$ for psid in $(pgrep vpp)
$ do
$     for tid in $(ps -Lo tid --pid $psid | grep -v TID)
$     do
$         echo $tid
$     done
$ done
```

Or:

```
$ cat /proc/`pidof vpp`/task/*/stat | awk '{print $1" "$2" "$39}'
```

Applying Round-robin scheduling with highest priority

```
$ for psid in $(pgrep vpp)
$ do
$     for tid in $(ps -Lo tid --pid $psid | grep -v TID)
$     do
$         chrt -r -p 1 $tid
$     done
$ done
```

More information about Linux CFS can be found in: Sched manual pages[66].

Host IRQ affinity

Changing the default pinning of every IRQ to core 0. (Same does apply on both guest VM and host OS)

```
$ for l in `ls /proc/irq`; do echo 1 | sudo tee /proc/irq/$l/smp_affinity; done
```

Host RCU affinity

Changing the default pinning of RCU to core 0. (Same does apply on both guest VM and host OS)

[66] http://man7.org/linux/man-pages/man7/sched.7.html

```
$ for i in `pgrep rcu[^c]` ; do sudo taskset -pc 0 $i ; done
```

Host Writeback affinity

Changing the default pinning of writebacks to core 0. (Same does apply on both guest VM and host OS)

```
$ echo 1 | sudo tee /sys/bus/workqueue/devices/writeback/cpumask
```

2.7.4 DUT Configuration - VPP

VPP Version

VPP-18.01.2 release

VPP Compile Parameters

FD.io VPP compile job[67]

VPP Install Parameters

```
$ dpkg -i --force-all vpp*
```

VPP Startup Configuration

VPP startup configuration changes per test case with different settings for CPU cores, rx-queues and no-multi-seg parameter. Startup config is aligned with applied test case tag:

Tagged by **1T1C**

```
unix
{
  cli-listen localhost:5002
  log /tmp/vpe.log
  nodaemon
}
cpu
{
  corelist-workers 2
  main-core 1
}
ip4
{
  heap-size "4G"
}
ip6
{
  heap-size "4G"
  hash-buckets "2000000"
}
plugins
{
  plugin pppoe_plugin.so { disable }
  plugin kubeproxy_plugin.so { disable }
  plugin ioam_plugin.so { disable }
  plugin ila_plugin.so { disable }
  plugin stn_plugin.so { disable }
  plugin acl_plugin.so { disable }
  plugin l2e_plugin.so { disable }
  plugin sixrd_plugin.so { disable }
  plugin nat_plugin.so { disable }
  plugin ixge_plugin.so { disable }
```

(continues on next page)

[67] https://jenkins.fd.io/view/vpp/job/vpp-merge-1801-ubuntu1604/

(continued from previous page)

```
  plugin lb_plugin.so { disable }
  plugin memif_plugin.so { disable }
  plugin gtpu_plugin.so { disable }
  plugin flowprobe_plugin.so { disable }
}
heapsize "4G"
dpdk
{
  dev 0000:88:00.1
  dev 0000:88:00.0
  no-multi-seg
  dev default
  {
    num-rx-desc 2048
    num-rx-queues 1
    num-tx-desc 2048
  }
  socket-mem "1024,1024"
  no-tx-checksum-offload
}
```

Tagged by **2T2C**

```
unix
{
  cli-listen localhost:5002
  log /tmp/vpe.log
  nodaemon
}
cpu
{
  corelist-workers 2,3
  main-core 1
}
ip4
{
  heap-size "4G"
}
ip6
{
  heap-size "4G"
  hash-buckets "2000000"
}
plugins
{
  plugin pppoe_plugin.so { disable }
  plugin kubeproxy_plugin.so { disable }
  plugin ioam_plugin.so { disable }
  plugin ila_plugin.so { disable }
  plugin stn_plugin.so { disable }
  plugin acl_plugin.so { disable }
  plugin l2e_plugin.so { disable }
  plugin sixrd_plugin.so { disable }
  plugin nat_plugin.so { disable }
  plugin ixge_plugin.so { disable }
  plugin lb_plugin.so { disable }
  plugin memif_plugin.so { disable }
  plugin gtpu_plugin.so { disable }
  plugin flowprobe_plugin.so { disable }
}
heapsize "4G"
```

(continues on next page)

(continued from previous page)

```
dpdk
{
  dev 0000:88:00.1
  dev 0000:88:00.0
  no-multi-seg
  dev default
  {
    num-rx-desc 2048
    num-rx-queues 1
    num-tx-desc 2048
  }
  socket-mem "1024,1024"
  no-tx-checksum-offload
}
```

Tagged by **4T4C**

```
unix
{
  cli-listen localhost:5002
  log /tmp/vpe.log
  nodaemon
}
cpu
{
  corelist-workers 2,3,4,5
  main-core 1
}
ip4
{
  heap-size "4G"
}
ip6
{
  heap-size "4G"
  hash-buckets "2000000"
}
plugins
{
  plugin pppoe_plugin.so { disable }
  plugin kubeproxy_plugin.so { disable }
  plugin ioam_plugin.so { disable }
  plugin ila_plugin.so { disable }
  plugin stn_plugin.so { disable }
  plugin acl_plugin.so { disable }
  plugin l2e_plugin.so { disable }
  plugin sixrd_plugin.so { disable }
  plugin nat_plugin.so { disable }
  plugin ixge_plugin.so { disable }
  plugin lb_plugin.so { disable }
  plugin memif_plugin.so { disable }
  plugin gtpu_plugin.so { disable }
  plugin flowprobe_plugin.so { disable }
}
heapsize "4G"
dpdk
{
  dev 0000:88:00.1
  dev 0000:88:00.0
  no-multi-seg
  dev default
```

(continues on next page)

(continued from previous page)

```
  {
    num-rx-desc 2048
    num-rx-queues 2
    num-tx-desc 2048
  }
  socket-mem "1024,1024"
  no-tx-checksum-offload
}
```

2.7.5 TG Configuration - TRex

TG Version

TRex v2.35

DPDK version

DPDK v17.11

TG Build Script used

TRex intallation[68]

TG Startup Configuration

```
$ cat /etc/trex_cfg.yaml
- port_limit      : 2
  version         : 2
  interfaces      : ["0000:0d:00.0","0000:0d:00.1"]
  port_info       :
    - dest_mac        :   [0x3c,0xfd,0xfe,0x9c,0xee,0xf5]
      src_mac         :   [0x3c,0xfd,0xfe,0x9c,0xee,0xf4]
    - dest_mac        :   [0x3c,0xfd,0xfe,0x9c,0xee,0xf4]
      src_mac         :   [0x3c,0xfd,0xfe,0x9c,0xee,0xf5]
```

TG Startup Command

```
$ sh -c 'cd <t-rex-install-dir>/scripts/ && sudo nohup ./t-rex-64 -i -c 7 --iom 0 > /tmp/trex.log 2>
↪&1 &'> /dev/null
```

TG common API - pointer to driver

TRex driver[69]

2.8 Documentation

2.8.1 Container Orchestration in CSIT

Overview

Linux Containers

Linux Containers is an OS-level virtualization method for running multiple isolated Linux systems (containers) on a compute host using a single Linux kernel. Containers rely on Linux kernel cgroups functionality for controlling usage of shared system resources (i.e. CPU, memory, block I/O, network) and for

[68] https://git.fd.io/csit/tree/resources/tools/trex/trex_installer.sh?h=rls1801_2
[69] https://git.fd.io/csit/tree/resources/tools/trex/trex_stateless_profile.py?h=rls1801_2

namespace isolation. The latter enables complete isolation of applications' view of operating environment, including process trees, networking, user IDs and mounted file systems.

LXC combine kernel's cgroups and support for isolated namespaces to provide an isolated environment for applications. Docker does use LXC as one of its execution drivers, enabling image management and providing deployment services. More information in *[lxc]* (page 234), *[lxc-namespace]* (page 234) and *[stgraber]* (page 234).

Linux containers can be of two kinds: privileged containers and unprivileged containers.

Unprivileged Containers

Running unprivileged containers is the safest way to run containers in a production environment. From LXC 1.0 one can start a full system container entirely as a user, allowing to map a range of UIDs on the host into a namespace inside of which a user with UID 0 can exist again. In other words an unprivileged container does mask the userid from the host, making it impossible to gain a root access on the host even if a user gets root in a container. With unprivileged containers, non-root users can create containers and will appear in the container as the root, but will appear as userid <non-zero> on the host. Unprivileged containers are also better suited to supporting multi-tenancy operating environments. More information in *[lxc-security]* (page 234) and *[stgraber]* (page 234).

Privileged Containers

Privileged containers do not mask UIDs, and container UID 0 is mapped to the host UID 0. Security and isolation is controlled by a good configuration of cgroup access, extensive AppArmor profile preventing the known attacks as well as container capabilities and SELinux. Here a list of applicable security control mechanisms:

- Capabilities - keep (whitelist) or drop (blacklist) Linux capabilities, *[capabilities]* (page 234).
- Control groups - cgroups, resource bean counting, resource quotas, access restrictions, *[cgroup1]* (page 234), *[cgroup2]* (page 234).
- AppArmor - apparmor profiles aim to prevent any of the known ways of escaping a container or cause harm to the host, *[apparmor]* (page 234).
- SELinux - Security Enhanced Linux is a Linux kernel security module that provides similar function to AppArmor, supporting access control security policies including United States Department of Defense–style mandatory access controls. Mandatory access controls allow an administrator of a system to define how applications and users can access different resources such as files, devices, networks and inter- process communication, *[selinux]* (page 234).
- Seccomp - secure computing mode, enables filtering of system calls, *[seccomp]* (page 234).

More information in *[lxc-security]* (page 234) and *[lxc-sec-features]* (page 234).

Linux Containers in CSIT

CSIT is using Privileged Containers as the `sysfs` is mounted with RW access. Sysfs is required to be mounted as RW due to VPP accessing **`/sys/bus/pci/drivers/uio_pci_generic/unbind`**. This is not the case of unprivileged containers where `sysfs` is mounted as read-only.

Orchestrating Container Lifecycle Events

Following Linux container lifecycle events need to be addressed by an orchestration system:

1. Acquire - acquiring/downloading existing container images via **`docker pull`** or **`lxc-create -t download`**.

2. Build - building a container image from scratch or another container image via **docker build <dockerfile/composefile>** or customizing LXC templates in 'https://github.com/lxc/lxc/tree/master/templates'_
3. (Re-)Create - creating a running instance of a container application from anew, or re-creating one that failed. A.k.a. (re-)deploy via **docker run** or **lxc-start**
4. Execute - execute system operations within the container by attaching to running container. THis is done by **lxc-attach** or **docker exec**
5. Distribute - distributing pre-built container images to the compute nodes. Currently not implemented in CSIT.

Container Orchestration Systems Used in CSIT

Current CSIT testing framework integrates following Linux container orchestration mechanisms:

- LXC/Docker for complete VPP container lifecycle control.
- Combination of Kubernetes (container orchestration), Docker (container images) and Ligato (container networking).

LXC

LXC is the well-known and heavily tested low-level Linux container runtime *[lxc-source]* (page 234), that provides a userspace interface for the Linux kernel containment features. With a powerful API and simple tools, LXC enables Linux users to easily create and manage system or application containers. LXC uses following kernel features to contain processes:

- Kernel namespaces: ipc, uts, mount, pid, network and user.
- AppArmor and SELinux security profiles.
- Seccomp policies.
- Chroot.
- Cgroups.

CSIT uses LXC runtime and LXC usertools to test VPP data plane performance in a range of virtual networking topologies.

Known Issues

- Current CSIT restriction: only single instance of lxc runtime due to the cgroup policies used in CSIT. There is plan to add the capability into code to create cgroups per container instance to address this issue. This sort of functionality is better supported in LXC 2.1 but can be done is current version as well.
- CSIT code is currently using cgroup to control the range of CPU cores the LXC container runs on. VPP thread pinning is defined vpp startup.conf.

Docker

Docker builds on top of Linux kernel containment features, and offers a high-level tool for wrapping the processes, maintaining and executing them in containers *[docker]* (page 234). Currently it using *runc* a CLI tool for spawning and running containers according to the OCI specification[70]

A Docker container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, settings.

[70] https://www.opencontainers.org/

CSIT uses Docker to manage the maintenance and execution of containerized applications used in CSIT performance tests.

- Data plane thread pinning to CPU cores - Docker CLI and/or Docker configuration file controls the range of CPU cores the Docker image must run on. VPP thread pinning defined vpp startup.conf.

Kubernetes

Kubernetes *[k8s-doc]* (page 234), or K8s, is a production-grade container orchestration platform for automating the deployment, scaling and operating application containers. Kubernetes groups containers that make up an application into logical units, pods, for easy management and discovery. K8s pod definitions including compute resource allocation is provided in .yaml files.

CSIT uses K8s and its infrastructure components like etcd to control all phases of container based virtualized network topologies.

Ligato

Ligato *[ligato]* (page 234) is an open-source project developing a set of cloud-native tools for orchestrating container networking. Ligato integrates with FD.io VPP using goVPP *[govpp]* (page 234) and vpp-agent *[vpp-agent]* (page 235).

Known Issues

- Currently using a separate LF Jenkins job for building csit-centric prod_vpp_agent docker images vs. dockerhub/ligato ones.

Implementation

CSIT container orchestration is implemented in CSIT Level-1 keyword Python libraries following the Builder design pattern. Builder design pattern separates the construction of a complex object from its representation, so that the same construction process can create different representations e.g. LXC, Docker, other.

CSIT Robot Framework keywords are then responsible for higher level lifecycle control of of the named container groups. One can have multiple named groups, with 1..N containers in a group performing different role/functionality e.g. NFs, Switch, Kafka bus, ETCD datastore, etc. ContainerManager class acts as a Director and uses ContainerEngine class that encapsulate container control.

Current CSIT implementation is illustrated using UML Class diagram:

1. Acquire
2. Build
3. (Re-)Create
4. Execute

```
+---------------------------------------------------------------------+
|              RF Keywords (high level lifecycle control)             |
+---------------------------------------------------------------------+
| Construct VNF containers on all DUTs                                |
| Acquire all '${group}' containers                                   |
| Create all '${group}' containers                                    |
| Install all '${group}' containers                                   |
| Configure all '${group}' containers                                 |
| Stop all '${group}' containers                                      |
| Destroy all '${group}' containers                                   |
+----------------+----------------------------------------------------+
                 |  1
```

(continues on next page)

(continued from previous page)

```
                  |
                  |  1..N
+-----------------v-----------------+        +--------------------------+
|          ContainerManager         |        |  ContainerEngine         |
+-----------------------------------+        +--------------------------+
| __init()__                        |        | __init(node)__           |
| construct_container()             |        | acquire(force)           |
| construct_containers()            |        | create()                 |
| acquire_all_containers()          |        | stop()                   |
| create_all_containers()           | 1    1 | destroy()                |
| execute_on_container()            <>-------| info()                   |
| execute_on_all_containers()       |        | execute(command)         |
| install_vpp_in_all_containers()   |        | system_info()            |
| configure_vpp_in_all_containers() |        | install_supervisor()     |
| stop_all_containers()             |        | install_vpp()            |
| destroy_all_containers()          |        | restart_vpp()            |
+-----------------------------------+        | create_vpp_exec_config() |
                                             | create_vpp_startup_config|
                                             | is_container_running()   |
                                             | is_container_present()   |
                                             | _configure_cgroup()      |
                                             +------------^-------------+
                                                          |
                                                          |
                                                          |
                                                +---------+---------+
                                                |                   |
                                          +-----+-------+     +-----+-------+
                                          |     LXC     |     |    Docker   |
                                          +-------------+     +-------------+
                                          | (inherinted) |     | (inherinted) |
                                          +-----+-------+     +-----+-------+
                                                |                   |
                                                +---------+---------+
                                                          |
                                                          | constructs
                                                          |
                                                +---------v---------+
                                                |     Container     |
                                                +-------------------+
                                                | __getattr__(a)    |
                                                | __setattr__(a, v) |
                                                +-------------------+
```

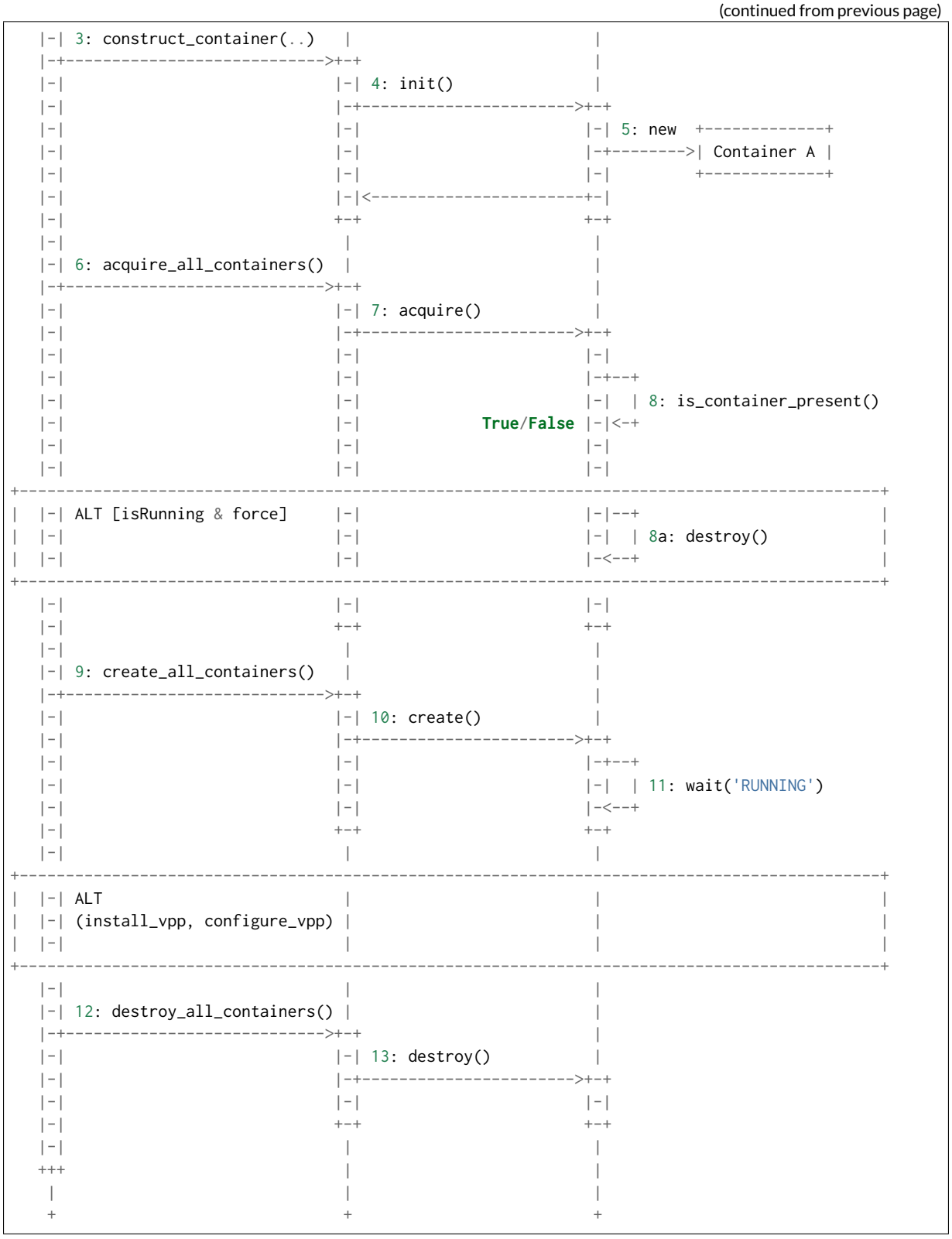
Sequentional diagram that illustrates the creation of a single container.

```
Legend:
   e  = engine [Docker|LXC]
   .. = kwargs (variable number of keyword argument)

+-------+                 +------------------+       +-----------------+
| RF KW |                 | ContainerManager |       | ContainerEngine |
+---+---+                 +--------+---------+       +--------+--------+
    |                              |                          |
    |  1: new ContainerManager(e)  |                          |
   +-+----------------------------->+-+                       |
   |-|                              |-| 2: new ContainerEngine  |
   |-|                              |-+----------------------->+-+
   |-|                              |-|                        |-|
   |-|                              +-+                        +-+
   |-|                                |                          |
```

(continues on next page)

(continued from previous page)



Container Data Structure

Container is represented in Python L1 library as a separate Class with instance variables and no methods except overriden `__getattr__` and `__setattr__`. Instance variables are assigned to container dynamically during the `construct_container(**kwargs)` call and are passed down from the RF keyword.

Usage example:

```
| Construct VNF containers on all DUTs
| | [Arguments] | ${technology} | ${image} | ${cpu_count}=${1} | ${count}=${1}
| | ...
| | ${group}= | Set Variable | VNF
| | ${guest_dir}= | Set Variable | /mnt/host
| | ${host_dir}= | Set Variable | /tmp
| | ${skip_cpus}= | Evaluate | ${vpp_cpus}+${system_cpus}
| | Import Library | resources.libraries.python.ContainerUtils.ContainerManager
| | ... | engine=${technology} | WITH NAME | ${group}
| | ${duts}= | Get Matches | ${nodes} | DUT*
| | :FOR | ${dut} | IN | @{duts}
| | | {env}= | Create List | LC_ALL="en_US.UTF-8"
| | | ... | DEBIAN_FRONTEND=noninteractive | ETCDV3_ENDPOINTS=172.17.0.1:2379
| | | ${cpu_node}= | Get interfaces numa node | ${nodes['${dut}']}
| | | ... | ${dut1_if1} | ${dut1_if2}
| | | Run Keyword | ${group}.Construct containers
| | | ... | name=${dut}_${group}
| | | ... | node=${nodes['${dut}']}
| | | ... | host_dir=${host_dir}
| | | ... | guest_dir=${guest_dir}
| | | ... | image=${image}
| | | ... | cpu_count=${cpu_count}
| | | ... | cpu_skip=${skip_cpus}
| | | ... | smt_used=${False}
| | | ... | cpuset_mems=${cpu_node}
| | | ... | cpu_shared=${False}
| | | ... | env=${env}
```

Mandatory parameters to create standalone container are: `node`, `name`, `image` *[image-var]* (page 235), `cpu_count`, `cpu_skip`, `smt_used`, `cpuset_mems`, `cpu_shared`.

There is no parameters check functionality. Passing required arguments is in coder responsibility. All the above parameters are required to calculate the correct cpu placement. See documentation for the full reference.

Kubernetes

Kubernetes is implemented as separate library `KubernetesUtils.py`, with a class with the same name. This utility provides an API for L2 Robot Keywords to control `kubectl` installed on each of DUTs. One time initialization script, `resources/libraries/bash/k8s_setup.sh` does reset/init kubectl, applies Calico v2.6.3 and initializes the `csit` namespace. CSIT namespace is required to not to interfere with existing setups and it further simplifies apply/get/delete Pod/ConfigMap operations on SUTs.

Kubernetes utility is based on YAML templates to avoid crafting the huge YAML configuration files, what would lower the readability of code and requires complicated algorithms. The templates can be found in `resources/templates/kubernetes` and can be leveraged in the future for other separate tasks.

Two types of YAML templates are defined:

- Static - do not change between deployments, that is infrastructure containers like Kafka, Calico, ETCD.
- Dynamic - per test suite/case topology YAML files e.g. SFC_controller, VNF, VSWITCH.

Making own python wrapper library of `kubectl` instead of using the official Python package allows to control and deploy environment over the SSH library without the need of using isolated driver running on each of DUTs.

Ligato

Ligato integration does require to compile the `vpp-agent` tool and build the bundled Docker image. Compilation of `vpp-agent` depends on specific VPP. In `ligato/vpp-agent` repository there are well prepared scripts for building the Docker image. Building docker image is possible via series of commands:

```
git clone https://github.com/ligato/vpp-agent
cd vpp_agent/docker/dev_vpp_agent
sudo docker build -t dev_vpp_agent --build-arg AGENT_COMMIT=<agent commit id>\
    --build-arg VPP_COMMIT=<vpp commit id> --no-cache .
sudo ./shrink.sh
cd ../prod_vpp_agent
sudo ./build.sh
sudo ./shrink.sh
```

CSIT requires Docker image to include the desired VPP version (per patch testing, nightly testing, on demand testing).

The entire build process of building `dev_vpp_agent` image heavily depends on internet connectivity and also takes a significant amount of time (~1-1.5h based on internet bandwidth and allocated resources). The optimal solution would be to build the image on jenkins slave, transfer the Docker image to DUTs and execute separate suite of tests.

To adress the amount of time required to build `dev_vpp_agent` image, we can pull existing specific version of `dev_vpp_agent` and exctract the `vpp-agent` from it.

We created separate sets of Jenkins jobs, that will be executing following:

1. Clone latest CSIT and Ligato repositaries.
2. Pull specific version of `dev_vpp_agent` image from Dockerhub.
3. Extract VPP API (from `.deb` package) and copy into `dev_vpp_agent` image
4. Rebuild vpp-agent and extract outside image.
5. Build `prod_vpp_image` Docker image from `dev_vpp_agent` image.
6. Transfer `prod_vpp_agent` image to DUTs.
7. Execute subset of performance tests designed for Ligato testing.

```
+-----------------------------------------------+
|                  ubuntu:16.04                 <-----| Base image on Dockerhub
+-----------------------^-----------------------+
                        |
                        |
+-----------------------+-----------------------+
|               ligato/dev_vpp_agent            <------| Pull this image from
+-----------------------^-----------------------+      | Dockerhub ligato/dev_vpp_agent:<version>
                        |
                        | Rebuild and extract agent.tar.gz from dev_vpp_agent
+-----------------------+-----------------------+
|                 prod_vpp_agent                <------| Build by passing own
+-----------------------------------------------+      | vpp.tar.gz (from nexus
                                                       | or built by JJB) and
                                                       | agent.tar.gz extracted
                                                       | from ligato/dev_vpp_agent
```

Approximate size of vnf-agent docker images:

```
REPOSITORY            TAG       IMAGE ID        CREATED         SIZE
dev-vpp-agent         latest    78c53bd57e2     6 weeks ago     9.79GB
prod_vpp_agent        latest    f68af5afe601    5 weeks ago     443MB
```

In CSIT we need to create separate performance suite under `tests/kubernetes/perf` which contains modified Suite setup in comparison to standard perf tests. This is due to reason that VPP will act as vswitch in Docker image and not as standalone installed service.

Tested Topologies

Listed CSIT container networking test topologies are defined with DUT containerized VPP switch forwarding packets between NF containers. Each NF container runs their own instance of VPP in L2XC configuration.

Following container networking topologies are tested in CSIT rls1801_2:

- LXC topologies:
  - eth-l2xcbase-eth-2memif-1lxc.
  - eth-l2bdbasemaclrn-eth-2memif-1lxc.
- Docker topologies:
  - eth-l2xcbase-eth-2memif-1docker.
- Kubernetes/Ligato topologies:
  - eth-1drcl2bdbasemaclrn-eth-2memif-1drcl2xc-1paral
  - eth-1drcl2bdbasemaclrn-eth-2memif-2drcl2xc-1horiz
  - eth-1drcl2bdbasemaclrn-eth-2memif-4drcl2xc-1horiz
  - eth-1drcl2bdbasemaclrn-eth-4memif-2drcl2xc-1chain
  - eth-1drcl2bdbasemaclrn-eth-8memif-4drcl2xc-1chain
  - eth-1drcl2xcbase-eth-2memif-1drcl2xc-1paral
  - eth-1drcl2xcbase-eth-2memif-2drcl2xc-1horiz
  - eth-1drcl2xcbase-eth-2memif-4drcl2xc-1horiz
  - eth-1drcl2xcbase-eth-4memif-2drcl2xc-1chain
  - eth-1drcl2xcbase-eth-8memif-4drcl2xc-1chain

References

2.8.2 VPP Performance Tests

CSIT VPP Performance Tests Documentation[87] contains detailed functional description and input parameters for each test case.

[87] https://docs.fd.io/csit/rls1801_2/doc/tests.vpp.perf.html

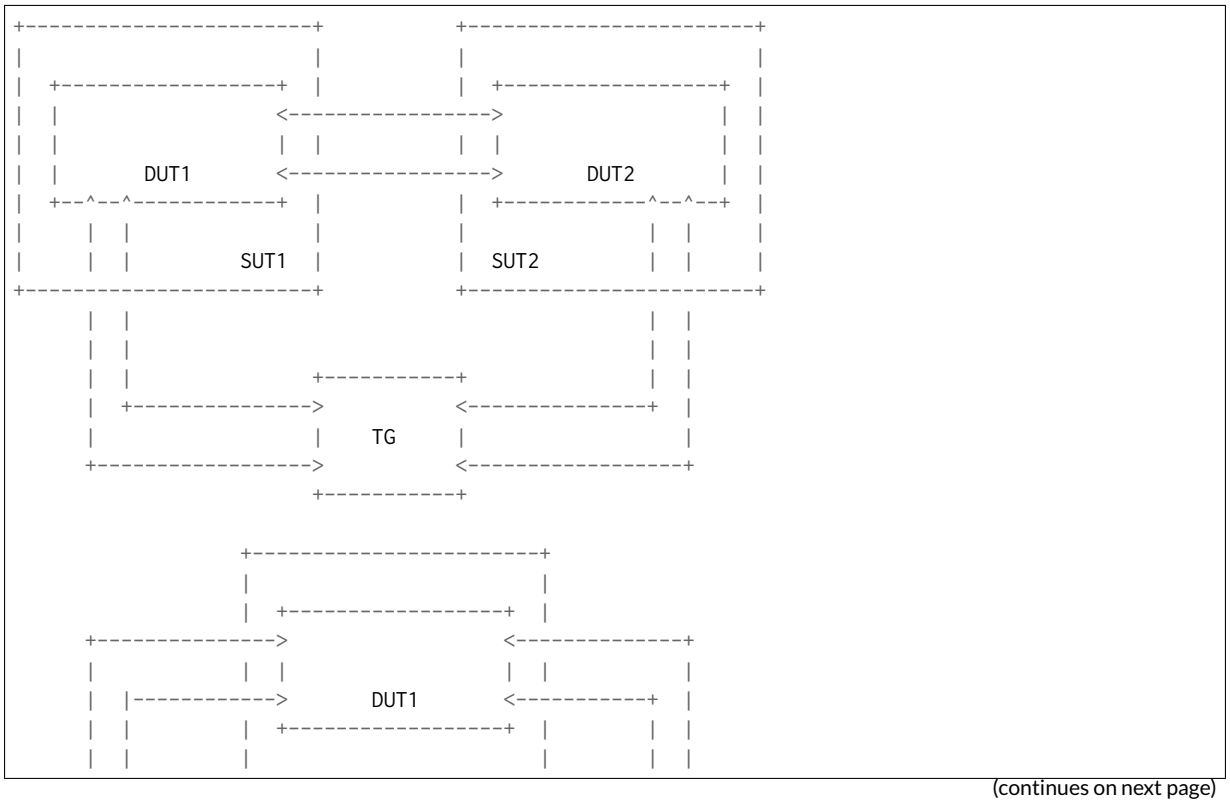
CHAPTER 3

VPP Functional Tests

3.1 Overview

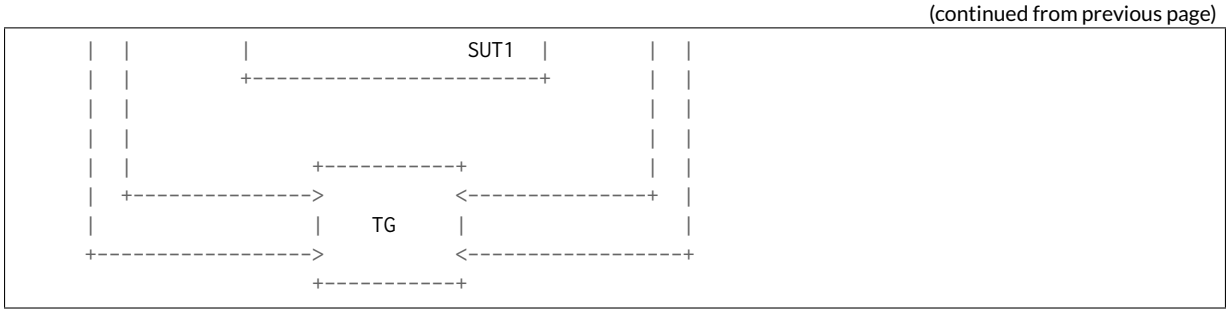
3.1.1 Tested Virtual Topologies

CSIT VPP functional tests are executed on virtualized topologies created using VIRL (Virtual Internet Routing Lab) simulation platform contributed by Cisco. VIRL runs on physical baremetal servers hosted by LF FD.io project. Majority of the tests are executed in the three node logical test topology - Traffic Generator (TG) node and two Systems Under Test (SUT) nodes connected in a loop. Some tests use two node logical test topology - TG node and SUT1 node. Both logical test topologies are shown in the figures below.:



(continues on next page)

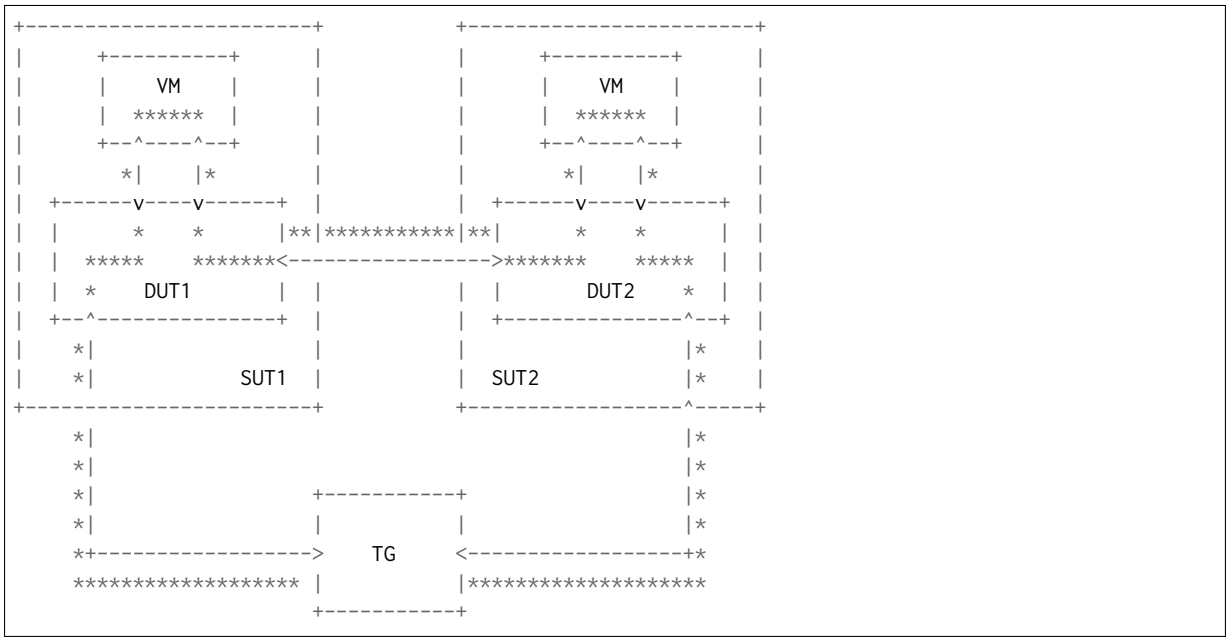
(continued from previous page)



SUT1 and SUT2 are two VMs (Ubuntu or Centos, depending on the test suite), TG is a Traffic Generator (TG, another Ubuntu VM). SUTs run VPP SW application in Linux user-mode as a Device Under Test (DUT) within the VM. TG runs Scapy SW application as a packet Traffic Generator. Logical connectivity between SUTs and to TG is provided using virtual NICs using VMs' virtio driver.

Virtual testbeds are created on-demand whenever a verification job is started (e.g. triggered by the gerrit patch submission) and destroyed upon completion of all functional tests. Each node is a Virtual Machine and each connection that is drawn on the diagram is available for use in any test case. During the test execution, all nodes are reachable thru the Management network connected to every node via dedicated virtual NICs and virtual links (not shown above for clarity).

For the test cases that require DUT (VPP) to communicate with VM over the vhost-user interfaces, a nested VM is created on SUT1 and/or SUT2 for the duration of these particular test cases only. DUT (VPP) test topology with VM is shown in the figure below including the applicable packet flow thru the VM (marked in the figure with ***).:



3.1.2 Functional Tests Coverage

Following VPP functional test areas are covered in the CSIT rls1801_2 with results listed in this report:

- **DHCP - Client and Proxy** - Dynamic Host Control Protocol Client and Proxy for IPv4, IPv6.
- **GRE Overlay Tunnels** - Generic Routing Encapsulation for IPv4.
- **L2BD Ethernet Switching** - L2 Bridge-Domain switched-forwarding for untagged Ethernet, dot1q and dot1ad tagged.
- **L2XC Ethernet Switching** - L2 Cross-Connect switched-forwarding for untagged Ethernet, dot1q and dot1ad tagged.

- **LISP Overlay Tunnels** - Locator/ID Separation Protocol overlay tunnels and locator/id mapping control.
- **Softwire Tunnels** - IPv4-in-IPv6 softwire tunnels.
- **Cop Address Security** - address white-list and black-list filtering for IPv4, IPv6.
- **IPSec - Tunnels and Transport** - IPSec tunnel and transport modes.
- **IPv6 Routed-Forwarding** - IPv6 routed-forwarding, NS/ND, RA, ICMPv6.
- **uRPF Source Security** - unicast Reverse Path Forwarding security.
- **Tap Interface** - baseline Linux tap interface tests.
- **Telemetry - IPFIX and SPAN** - IPFIX netflow statistics and SPAN port mirroring.
- **VRF Routed-Forwarding** - multi-context IPVPN routed-forwarding for IPv4, IPv6.
- **iACL Security** - ingress Access Control List security for IPv4, IPv6, MAC.
- **IPv4 Routed-Forwarding** - IPv4 routed-forwarding, RPF, ARP, Proxy ARP, ICMPv4.
- **QoS Policer Metering** - ingress packet rate measuring and marking for IPv4, IPv6.
- **VLAN Tag Translation** - L2 VLAN tag translation 2to2, 2to1, 1to2, 1to1.
- **VXLAN Overlay Tunnels** - VXLAN tunneling for L2-over-IP, for IPv4, IPv6.

3.1.3 Functional Tests Naming

CSIT rls1801_2 follows a common structured naming convention for all performance and system functional tests, introduced in CSIT rls1710.

The naming should be intuitive for majority of the tests. Complete description of CSIT test naming convention is provided on CSIT test naming wiki page[88].

Here few illustrative examples of the new naming usage for functional test suites:

1. **Physical port to physical port - a.k.a. NIC-to-NIC, Phy-to-Phy, P2P**
   - *eth2p-ethip4-ip4base-func.robot* => 2 ports of Ethernet, IPv4 baseline routed forwarding, functional tests.
2. **Physical port to VM (or VM chain) to physical port - a.k.a. NIC2VM2NIC, P2V2P, NIC2VMchain2NIC, P2V2V2P**
   - *eth2p-ethip4vxlan-l2bdbasemaclrn-eth-2vhost-1vm-func.robot* => 2 ports of Ethernet, IPv4 VXLAN Ethernet, L2 bridge-domain switching to/from two vhost interfaces and one VM, functional tests.

3.2 CSIT Release Notes

3.2.1 Changes in CSIT rls1801_2

1. VPP functional test framework changes:
   - improved stability of tests with nested-VM and QEMU;
2. Code updates and optimizations in CSIT functional framework:
   - IPSecSW tests - Software based IPSec encryption with CBC-SHA1 and CBC-SHA ciphers, in combination with IPv4 routed-forwarding;

[88] https://wiki.fd.io/view/CSIT/csit-test-naming

3.2.2 Known Issues

Here is the list of known issues in CSIT rls1801_2 for VPP functional tests in VIRL:

#	Issue	Jira ID	Description
1	DHCPv4 client: Client responses to DHCPv4 OFFER sent with different XID.	CSIT-129	Client replies with DHCPv4 REQUEST message when received DHCPv4 OFFER message with different (wrong) XID.
2	Softwire - MAP-E: Incorrect calculation of IPv6 destination address when IPv4 prefix is 0.	CSIT-398	IPv6 destination address is wrongly calculated in case that IPv4 prefix is equal to 0 and IPv6 prefix is less than 40.
3	Softwire - MAP-E: Map domain is created when incorrect parameters provided.	CSIT-399	Map domain is created in case that the sum of suffix length of IPv4 prefix and PSID length is greater than EA bits length. IPv6 destination address contains bits writen with PSID over the EA-bit length when IPv4 packet is sent.
4	IPv6 RA: Incorrect IPv6 destination address in response to ICMPv6 Router Solicitation.	CSIT-409	Wrong IPv6 destination address (ff02::1) is used in ICMPv6 Router Advertisement packet sent as a response to received ICMPv6 Router Solicitation packet.
5	Vhost-user: QEMU reconnect does not work.	CSIT-565	Using QEMU 2.5.0 that does not support vhost-user reconnect. It requires moving CSIT VIRL environment to QEMU 2.7.0.

3.3 Test Environment

CSIT functional tests are currently executed in FD.IO VIRL testbed. The physical VIRL testbed infrastructure consists of three VIRL hosts:

- All hosts are Cisco UCS C240-M4 (2x Intel(R) Xeon(R) CPU E5-2699 v3 @2.30GHz, 18c, 512GB RAM)
- tb4-virl1:
  - Status: Production
  - OS: Ubuntu 16.04.2
  - STD server version 0.10.32.16
  - UWM server version 0.10.32.16
- tb4-virl2:
  - Status: Production
  - OS: Ubuntu 16.04.2
  - STD server version 0.10.32.16
  - UWM server version 0.10.32.16
- tb4-virl3:
  - Status: Testing
  - OS: Ubuntu 16.04.2
  - STD server version 0.10.32.19
  - UWM server version 0.10.32.19

Whenever a patch is submitted to gerrit for review, parallel VIRL simulations are started to reduce the time of execution of all functional tests. The number of parallel VIRL simulations is equal to number of test groups defined by TEST_GROUPS variable in `csit/bootstrap.sh` file. The VIRL host to run VIRL simulation is selected based on least load algorithm per VIRL simulation.

Every VIRL simulation uses the same three-node - Traffic Generator (TG node) and two Systems Under Test (SUT1 and SUT2) - "double-ring" topology. The appropriate pre-built VPP packages built by Jenkins for the patch under review are then installed on the two SUTs, along with their `/etc/vpp/startup.conf` file, in all VIRL simulations.

3.3.1 SUT Configuration - VIRL Guest VM

Configurations of the SUT VMs is defined in VIRL topologies directory[89]

- List of SUT VM interfaces::

```
<interface id="0" name="GigabitEthernet0/4/0"/>
<interface id="1" name="GigabitEthernet0/5/0"/>
<interface id="2" name="GigabitEthernet0/6/0"/>
<interface id="3" name="GigabitEthernet0/7/0"/>
```

- Number of 2MB hugepages: 1024
- Maximum number of memory map areas: 20000
- Kernel Shared Memory Max: 2147483648 (vm.nr_hugepages * 2 * 1024 * 1024)

3.3.2 SUT Configuration - VIRL Guest OS Linux

In CSIT terminology, the VM operating system for both SUTs that VPP-18.01.2 release has been tested with, is the following:

1. **Ubuntu VIRL image**

   This image implies Ubuntu 16.04.1 LTS, current as of yyyy-mm-dd (that is, package versions are those that would have been installed by a **`apt-get update`**, **`apt-get upgrade`** on that day), produced by CSIT disk image build scripts.

   The exact list of installed packages and their versions (including the Linux kernel package version) are included in VIRL images lists[90].

   A replica of this VM image can be built by running the **`build.sh`** script in CSIT repository.

2. **CentOS VIRL image**

   The Centos7.3 image is ready to be used but no tests running on it now. Corresponding Jenkins jobs are under preparation.

   The exact list of installed packages and their versions (including the Linux kernel package version) are included in VIRL images lists[91].

   A replica of this VM image can be built by running the **`build.sh`** script in CSIT repository.

3. **Nested VM image**

   In addition to the "main" VM image, tests which require VPP to communicate to a VM over a vhost-user interface, utilize a "nested" VM image.

   This "nested" VM is dynamically created and destroyed as part of a test case, and therefore the "nested" VM image is optimized to be small, lightweight and have a short boot time. The "nested"

[89] https://git.fd.io/csit/tree/resources/tools/virl/topologies/?h=rls1801_2
[90] https://git.fd.io/csit/tree/resources/tools/disk-image-builder/ubuntu/lists/?h=rls1801_2
[91] https://git.fd.io/csit/tree/resources/tools/disk-image-builder/ubuntu/lists/?h=rls1801_2

VM image is not built around any established Linux distribution, but is based on BuildRoot[92], a tool for building embedded Linux systems. Just as for the "main" image, scripts to produce an identical replica of the "nested" image are included in CSIT GIT repository, and the image can be rebuilt using the "build.sh" script at VIRL nested[93].

3.3.3 DUT Configuration - VPP

Every System Under Test runs VPP SW application in Linux user-mode as a Device Under Test (DUT) node.

DUT port configuration

Port configuration of DUTs is defined in topology file that is generated per VIRL simulation based on the definition stored in VIRL topologies directory[94].

Example of DUT nodes configuration::

```
DUT1:
    type: DUT
    host: "10.30.51.157"
    port: 22
    username: cisco
    honeycomb:
      user: admin
      passwd: admin
      port: 8183
      netconf_port: 2831
    priv_key: |
      -----BEGIN RSA PRIVATE KEY-----
      MIIEpgIBAAKCAQEAwUDlTpzSHpwLQotZOFS4AgcPNEWCnP1AB2hWFmvI+8Kah/gb
      v8ruZU9RqhPs56tyKzxbhvNkY4VbH5F1GilHZu3mLqzM4KfghMmaeMEjO1T7BYYd
      vuBfTvIluljfQ2vAlnYrDwn+ClxJk81m0pDgvrLEX4qVVh2sGh7UEkYy5r82DNa2
      4VjzPB1J/c8a9zP8FoZUhYIzF4FLvRMjUADpbMXgJMsGpaZLmz95ap0Eot7vb1Cc
      1LvF97iyBCrtIOSKRKA50ZhLGjMKmOwnYU+cP5718tbproDVi6VJOo7zeuXyetMs
      8YBl9kWblWG9BqP9jctFvsmi5G7hXgq1Y8u+DwIDAQABAoIBAQC/W4E0DHjLMny7
      0bvw2YKzD0Zw3fttdB94tkm4PdZv5MybooPnsAvLaXVV0hEdfVi5kzSWNl/LY/tN
      EP1BgGphc2QgB59/PPxGwFIjDCvUzlsZpynBHe+B/qh5ExNQcVvsIOqWI7DXlXaN
      0i/khOzmJ6HncRRah1spKimYRsaUUDskyg7q3QqMWVaqBbbMvLs/w7ZWd/zoDqCU
      MY/pCI6hkB3QbRo0OdiZLohphBl2ShABTwjvVyyKL5UA4jAEneJrhH5gWVLXnfgD
      p62W5CollKEYblC8mUkPxpP7Qo277zw3xaq+oktIZhc5SUEUd7nJZtNqVAHqkItW
      79VmpKyxAoGBAPfU+kqNPaTSvp+x1n5sn2SgipzDtgi9QqNmC4cjtrQQaaqI57SG
      OHw1jX8i7L2G1WvVtkHg060nlEVo5n65ffFOqeVBezLVJ7ghWI8U+oBiJJyQ4boD
      GJVNsoOSUQ0rtuGd9eVwfDk3ol9aCN0KK53oPfIYli29pyu4l095kg11AoGBAMef
      bPEMBI/2XmCPshLSwhGFl+dW8d+Klluj3CUQ/0vUlvma3dfBOYNsIwAgTP0iIUTg
      8DYE6KBCdPtxAUEI0YAEAKB9ry1tKR2NQEIPfslYytKErtwjAiqSi0heM6+zwEzu
      f54Z4oBhsMSL0jXoOMnu+NZzEc6EUdQeY4O+jhjzAoGBAIogC3dtjMPGKTP7+93u
      UE/XIioI8fWg9fj3sMka4IMu+pVvRCRbAjRH7JrFLkjbUyuMqs3Arnk9K+gbdQt/
      +m95Njtt6WoFXuPCwgbM3GidSmZwYT4454SfDzVBYScEDCNm1FuR+8ov9bFLDtGT
      D4gsngnGJj1MDFXTxZEn4nzZAoGBAKCg4WmpUPaCuXibyB+rZavxwsTNSn2lJ83/
      sYJGBhf/raiV/FLDUcM1vYg5dZnu37RsB/5/vqxOLZGyYd7x+Jo5HkQGPnKgNwhn
      g8BkdZIRF8uEJqxOo0ycdOU7n/2O93swIpKWo5LIiRPuqqzj+uZKnAL7vuVdxfaY
      qVz2daMPAoGBALgaaKa3voU/HO1PYLWIhFrBThyJ+BQSQ8OqrEzC8AnegWFxRAM8
      EqrzZXl7ACUuo1dH0Eipm41j2+BZWlQjiUgq5uj8+yzy+EU1ZRRyJcOKzbDACeuD
      BpWWSXGBI5G4CppeYLjMUHZpJYeX1USULJQd2c4crLJKb76E8gz3Z9kN
      -----END RSA PRIVATE KEY-----

    interfaces:
      port1:
        mac_address: "fa:16:3e:9b:89:52"
```

(continues on next page)

[92] https://buildroot.org/
[93] https://git.fd.io/csit/tree/resources/tools/disk-image-builder/nested/?h=rls1801_2
[94] https://git.fd.io/csit/tree/resources/tools/virl/topologies/?h=rls1801_2

(continued from previous page)

```
      pci_address: "0000:00:04.0"
      link: link1
    port2:
      mac_address: "fa:16:3e:7a:33:60"
      pci_address: "0000:00:05.0"
      link: link4
    port3:
      mac_address: "fa:16:3e:29:b7:ae"
      pci_address: "0000:00:06.0"
      link: link3
    port4:
      mac_address: "fa:16:3e:76:8d:ff"
      pci_address: "0000:00:07.0"
      link: link6
DUT2:
  type: DUT
  host: "10.30.51.156"
  port: 22
  username: cisco
  honeycomb:
    user: admin
    passwd: admin
    port: 8183
    netconf_port: 2831
  priv_key: |
    -----BEGIN RSA PRIVATE KEY-----
    MIIEpgIBAAKCAQEAwUDlTpzSHpwLQotZOFS4AgcPNEWCnP1AB2hWFmvI+8Kah/gb
    v8ruZU9RqhPs56tyKzxbhvNkY4VbH5F1GilHZu3mLqzM4KfghMmaeMEjO1T7BYYd
    vuBfTvIluljfQ2vAlnYrDwn+ClxJk81m0pDgvrLEX4qVVh2sGh7UEkYy5r82DNa2
    4VjzPB1J/c8a9zP8FoZUhYIzF4FLvRMjUADpbMXgJMsGpaZLmz95ap0Eot7vb1Cc
    1LvF97iyBCrtIOSKRKA50ZhLGjMKmOwnYU+cP5718tbproDVi6VJOo7zeuXyetMs
    8YBl9kWblWG9BqP9jctFvsmi5G7hXgq1Y8u+DwIDAQABAoIBAQC/W4E0DHjLMny7
    0bvw2YKzD0Zw3fttdB94tkm4PdZv5MybooPnsAvLaXVV0hEdfVi5kzSWNl/LY/tN
    EP1BgGphc2QgB59/PPxGwFIjDCvUzlsZpynBHe+B/qh5ExNQcVvsIOqWI7DXlXaN
    0i/khOzmJ6HncRRah1spKimYRsaUUDskyg7q3QqMWVaqBbbMvLs/w7ZWd/zoDqCU
    MY/pCI6hkB3QbRo0OdiZLohphBl2ShABTwjvVyyKL5UA4jAEneJrhH5gWVLXnfgD
    p62W5CollKEYblC8mUkPxpP7Qo277zw3xaq+oktIZhc5SUEUd7nJZtNqVAHqkItW
    79VmpKyxAoGBAPfU+kqNPaTSvp+x1n5sn2SgipzDtgi9QqNmC4cjtrQQaaqI57SG
    OHw1jX8i7L2G1WvVtkHg060nlEVo5n65ffFOqeVBezLVJ7ghWI8U+oBiJJyQ4boD
    GJVNsoOSUQ0rtuGd9eVwfDk3ol9aCN0KK53oPfIYli29pyu4l095kg11AoGBAMef
    bPEMBI/2XmCPshLSwhGFl+dW8d+Klluj3CUQ/0vUlvma3dfBOYNsIwAgTP0iIUTg
    8DYE6KBCdPtxAUEI0YAEAKB9ry1tKR2NQEIPfslYytKErtwjAiqSi0heM6+zwEzu
    f54Z4oBhsMSL0jXoOMnu+NZzEc6EUdQeY4O+jhjzAoGBAIogC3dtjMPGKTP7+93u
    UE/XIioI8fWg9fj3sMka4IMu+pVvRCRbAjRH7JrFLkjbUyuMqs3Arnk9K+gbdQt/
    +m95Njtt6WoFXuPCwgbM3GidSmZwYT4454SfDzVBYScEDCNm1FuR+8ov9bFLDtGT
    D4gsngnGJj1MDFXTxZEn4nzZAoGBAKCg4WmpUPaCuXibyB+rZavxwsTNSn2lJ83/
    sYJGBhf/raiV/FLDUcM1vYg5dZnu37RsB/5/vqxOLZGyYd7x+Jo5HkQGPnKgNwhn
    g8BkdZIRF8uEJqxOo0ycdOU7n/2O93swIpKWo5LIiRPuqqzj+uZKnAL7vuVdxfaY
    qVz2daMPAoGBALgaaKa3voU/HO1PYLWIhFrBThyJ+BQSQ8OqrEzC8AnegWFxRAM8
    EqrzZXl7ACUuo1dH0Eipm41j2+BZWlQjiUgq5uj8+yzy+EU1ZRRyJcOKzbDACeuD
    BpWWSXGBI5G4CppeYLjMUHZpJYeX1USULJQd2c4crLJKb76E8gz3Z9kN
    -----END RSA PRIVATE KEY-----

  interfaces:
    port1:
      mac_address: "fa:16:3e:ad:6c:7d"
      pci_address: "0000:00:04.0"
      link: link2
    port2:
      mac_address: "fa:16:3e:94:a4:99"
      pci_address: "0000:00:05.0"
```

(continues on next page)

(continued from previous page)

```
        link: link5
      port3:
        mac_address: "fa:16:3e:75:92:da"
        pci_address: "0000:00:06.0"
        link: link3
      port4:
        mac_address: "fa:16:3e:2c:b1:2a"
        pci_address: "0000:00:07.0"
        link: link6
```

VPP Version

VPP-18.01.2 release

VPP Installed Packages

```
$ dpkg -l vpp\*
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name           Version       Architecture Description
+++-==============-=============-============-=================================================
ii  vpp            17.07-release amd64        Vector Packet Processing--executables
ii  vpp-dbg        17.07-release amd64        Vector Packet Processing--debug symbols
ii  vpp-dev        17.07-release amd64        Vector Packet Processing--development support
ii  vpp-dpdk-dev   17.07-release amd64        Vector Packet Processing--development support
ii  vpp-dpdk-dkms  17.07-release amd64        DPDK 2.1 igb_uio_driver
ii  vpp-lib        17.07-release amd64        Vector Packet Processing--runtime libraries
ii  vpp-plugins    17.07-release amd64        Vector Packet Processing--runtime plugins
```

VPP Startup Configuration

VPP startup configuration is common for all test cases.

```
$ cat /etc/vpp/startup.conf
unix {
  nodaemon
  log /tmp/vpp.log
  full-coredump
}

api-trace {
  on
}

api-segment {
  gid vpp
}

cpu {
    ## In the VPP there is one main thread and optionally the user can create worker(s)
    ## The main thread and worker thread(s) can be pinned to CPU core(s) manually or automatically

    ## Manual pinning of thread(s) to CPU core(s)

    ## Set logical CPU core where main thread runs
    # main-core 1

    ## Set logical CPU core(s) where worker threads are running
    # corelist-workers 2-3,18-19

    ## Automatic pinning of thread(s) to CPU core(s)
```

(continues on next page)

(continued from previous page)

```
    ## Sets number of CPU core(s) to be skipped (1 ... N-1)
    ## Skipped CPU core(s) are not used for pinning main thread and working thread(s).
    ## The main thread is automatically pinned to the first available CPU core and worker(s)
    ## are pinned to next free CPU core(s) after core assigned to main thread
    # skip-cores 4

    ## Specify a number of workers to be created
    ## Workers are pinned to N consecutive CPU cores while skipping "skip-cores" CPU core(s)
    ## and main thread's CPU core
    # workers 2

    ## Set scheduling policy and priority of main and worker threads

    ## Scheduling policy options are: other (SCHED_OTHER), batch (SCHED_BATCH)
    ## idle (SCHED_IDLE), fifo (SCHED_FIFO), rr (SCHED_RR)
    # scheduler-policy fifo

    ## Scheduling priority is used only for "real-time policies (fifo and rr),
    ## and has to be in the range of priorities supported for a particular policy
    # scheduler-priority 50
}

dpdk {
    ## Change default settings for all intefaces
    # dev default {
        ## Number of receive queues, enables RSS
        ## Default is 1
        # num-rx-queues 3

        ## Number of transmit queues, Default is equal
        ## to number of worker threads or 1 if no workers treads
        # num-tx-queues 3

        ## Number of descriptors in transmit and receive rings
        ## increasing or reducing number can impact performance
        ## Default is 1024 for both rx and tx
        # num-rx-desc 512
        # num-tx-desc 512

        ## VLAN strip offload mode for interface
        ## Default is off
        # vlan-strip-offload on
    # }

    ## Whitelist specific interface by specifying PCI address
    # dev 0000:02:00.0

    ## Whitelist specific interface by specifying PCI address and in
    ## addition specify custom parameters for this interface
    # dev 0000:02:00.1 {
    #       num-rx-queues 2
    # }

    ## Change UIO driver used by VPP, Options are: uio_pci_generic, vfio-pci
    ## and igb_uio (default)
    # uio-driver uio_pci_generic

    ## Disable mutli-segment buffers, improves performance but
    ## disables Jumbo MTU support
    # no-multi-seg
```

(continues on next page)

(continued from previous page)

```
    ## Increase number of buffers allocated, needed only in scenarios with
    ## large number of interfaces and worker threads. Value is per CPU socket.
    ## Default is 32768
    # num-mbufs 128000

    ## Change hugepages allocation per-socket, needed only if there is need for
    ## larger number of mbufs. Default is 256M on each detected CPU socket
    # socket-mem 2048,2048
}
```

3.3.4 TG Configuration

Traffic Generator node is VM running the same OS Linux as SUTs. Ports of this VM are used as source (Tx) and destination (Rx) ports for the traffic.

Traffic scripts of test cases are executed on this VM.

TG VM configuration

Configuration of the TG VMs is defined in VIRL topologies directory[95].

/csit/resources/tools/virl/topologies/double-ring-nested.xenial.virl

- List of TG VM interfaces::

```
<interface id="0" name="eth1"/>
<interface id="1" name="eth2"/>
<interface id="2" name="eth3"/>
<interface id="3" name="eth4"/>
<interface id="4" name="eth5"/>
<interface id="5" name="eth6"/>
```

TG node port configuration

Port configuration of TG is defined in topology file that is generated per VIRL simulation based on the definition stored in VIRL topologies directory[96].

Example of TG node configuration::

```
TG:
    type: TG
    host: "10.30.51.155"
    port: 22
    username: cisco
    priv_key: |
      -----BEGIN RSA PRIVATE KEY-----
      MIIEpgIBAAKCAQEAwUDlTpzSHpwLQotZOFS4AgcPNEWCnP1AB2hWFmvI+8Kah/gb
      v8ruZU9RqhPs56tyKzxbhvNkY4VbH5F1GilHZu3mLqzM4KfghMmaeMEjO1T7BYYd
      vuBfTvIluljfQ2vAlnYrDwn+ClxJk81m0pDgvrLEX4qVVh2sGh7UEkYy5r82DNa2
      4VjzPB1J/c8a9zP8FoZUhYIzF4FLvRMjUADpbMXgJMsGpaZLmz95ap0Eot7vb1Cc
      1LvF97iyBCrtIOSKRKA50ZhLGjMKmOwnYU+cP5718tbproDVi6VJOo7zeuXyetMs
      8YBl9kWblWG9BqP9jctFvsmi5G7hXgq1Y8u+DwIDAQABAoIBAQC/W4E0DHjLMny7
      0bvw2YKzD0Zw3fttdB94tkm4PdZv5MybooPnsAvLaXVV0hEdfVi5kzSWNl/LY/tN
      EP1BgGphc2QgB59/PPxGwFIjDCvUzlsZpynBHe+B/qh5ExNQcVvsIOqWI7DXlXaN
      0i/khOzmJ6HncRRah1spKimYRsaUUDskyg7q3QqMWVaqBbbMvLs/w7ZWd/zoDqCU
      MY/pCI6hkB3QbRo0OdiZLohphBl2ShABTwjvVyyKL5UA4jAEneJrhH5gWVLXnfgD
      p62W5CollKEYblC8mUkPxpP7Qo277zw3xaq+oktIZhc5SUEUd7nJZtNqVAHqkItW
      79VmpKyxAoGBAPfU+kqNPaTSvp+x1n5sn2SgipzDtgi9QqNmC4cjtrQQaaqI57SG
```

(continues on next page)

[95] https://git.fd.io/csit/tree/resources/tools/virl/topologies/?h=rls1801_2
[96] https://git.fd.io/csit/tree/resources/tools/virl/topologies/?h=rls1801_2

(continued from previous page)

```
    OHw1jX8i7L2G1WvVtkHg060nlEVo5n65ffFOqeVBezLVJ7ghWI8U+oBiJJyQ4boD
    GJVNsoOSUQ0rtuGd9eVwfDk3ol9aCN0KK53oPfIYli29pyu4l095kg11AoGBAMef
    bPEMBI/2XmCPshLSwhGFl+dW8d+Klluj3CUQ/0vUlvma3dfBOYNsIwAgTP0iIUTg
    8DYE6KBCdPtxAUEI0YAEAKB9ry1tKR2NQEIPfslYytKErtwjAiqSi0heM6+zwEzu
    f54Z4oBhsMSL0jXoOMnu+NZzEc6EUdQeY4O+jhjzAoGBAIogC3dtjMPGKTP7+93u
    UE/XIioI8fWg9fj3sMka4IMu+pVvRCRbAjRH7JrFLkjbUyuMqs3Arnk9K+gbdQt/
    +m95Njtt6WoFXuPCwgbM3GidSmZwYT4454SfDzVBYScEDCNm1FuR+8ov9bFLDtGT
    D4gsngnGJj1MDFXTxZEn4nzZAoGBAKCg4WmpUPaCuXibyB+rZavxwsTNSn2lJ83/
    sYJGBhf/raiV/FLDUcM1vYg5dZnu37RsB/5/vqxOLZGyYd7x+Jo5HkQGPnKgNwhn
    g8BkdZIRF8uEJqxOo0ycdOU7n/2O93swIpKWo5LIiRPuqqzj+uZKnAL7vuVdxfaY
    qVz2daMPAoGBALgaaKa3voU/HO1PYLWIhFrBThyJ+BQSQ8OqrEzC8AnegWFxRAM8
    EqrzZXl7ACUuo1dH0Eipm41j2+BZWlQjiUgq5uj8+yzy+EU1ZRRyJcOKzbDACeuD
    BpWWSXGBI5G4CppeYLjMUHZpJYeX1USULJQd2c4crLJKb76E8gz3Z9kN
    -----END RSA PRIVATE KEY-----

  interfaces:
    port3:
      mac_address: "fa:16:3e:b9:e1:27"
      pci_address: "0000:00:06.0"
      link: link1
      driver: virtio-pci
    port4:
      mac_address: "fa:16:3e:e9:c8:68"
      pci_address: "0000:00:07.0"
      link: link4
      driver: virtio-pci
    port5:
      mac_address: "fa:16:3e:e8:d3:47"
      pci_address: "0000:00:08.0"
      link: link2
      driver: virtio-pci
    port6:
      mac_address: "fa:16:3e:cf:ca:58"
      pci_address: "0000:00:09.0"
      link: link5
      driver: virtio-pci
```

Traffic generator

Functional tests utilize Scapy as a traffic generator. There was used Scapy v2.3.1 for VPP-18.01.2 release tests.

3.4 Documentation

CSIT VPP Functional Tests Documentation[97] contains detailed functional description and input parameters for each test case.

[97] https://docs.fd.io/csit/rls1801_2/doc/tests.vpp.func.html

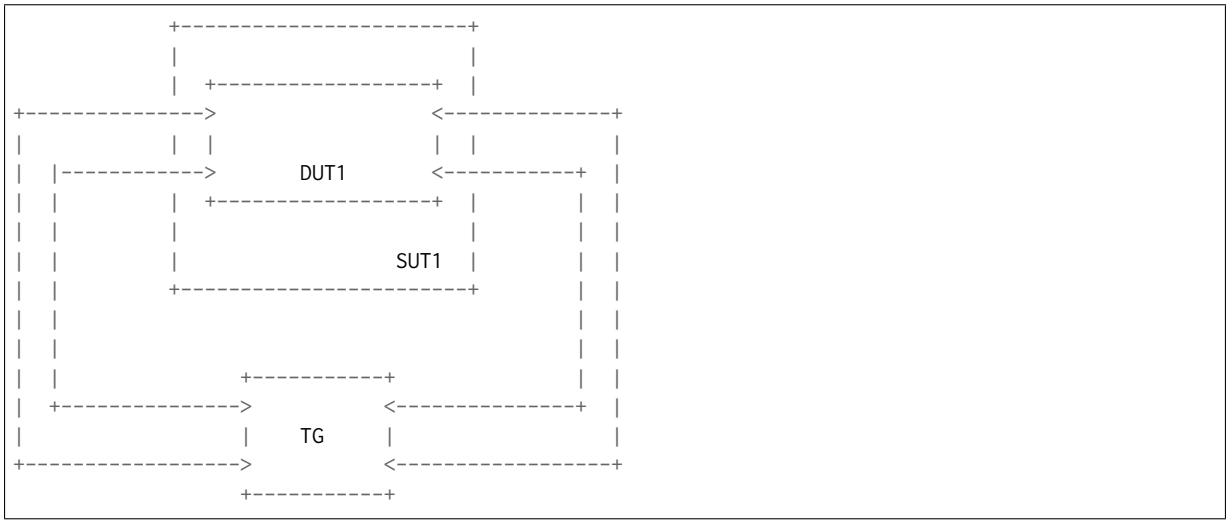
CHAPTER 4

Honeycomb Functional Tests

4.1 Overview

4.1.1 Tested Virtual Topologies

CSIT Honeycomb functional tests are executed on virtualized topologies created using VIRL simulation platform contributed by Cisco. VIRL runs on physical baremetal servers hosted by LF FD.io project. All tests are executed in two node logical test topology - Traffic Generator (TG) node and Systems Under Test (SUT1) node connected in a loop. Logical test topology is shown in the figure below.:



SUT1 is a VM (Ubuntu or Centos, depending on the test suite), TG is a Traffic Generator (TG, another Ubuntu VM). SUTs run Honeycomb and VPP SW applications in Linux user-mode as a Device Under Test (DUT) within the VM. TG runs Scapy SW application as a packet Traffic Generator. Logical connectivity between SUTs and to TG is provided using virtual NICs using VMs' virtio driver.

Virtual testbeds are created on-demand whenever a verification job is started (e.g. triggered by the gerrit patch submission) and destroyed upon completion of all functional tests. Each node is a Virtual Machine and each connection that is drawn on the diagram is available for use in any test case. During the test execution, all nodes are reachable thru the Management network connected to every node via dedicated virtual NICs and virtual links (not shown above for clarity).

4.1.2 Functional Tests Coverage

The following Honeycomb functional test areas are included in the CSIT rls1801_2 with results listed in this report:

- **Basic interface management** - CRUD for interface state, - ipv4/ipv6 address, ipv4 neighbor, MTU value. - Test case count: 14
- **L2BD** - CRUD for L2 Bridge-Domain, interface assignment. - Create up to two bridge domains with all implemented functions turned on. - (flooding, unknown-unicast flooding, forwarding, learning, arp-termination) - Assign up to two physical interfaces to a single bridge domain. - Remove interface assignments, remove bridge domains. - Test case count: 5
- **L2FIB** - CRD for L2-FIB entries. - Create 4 FIB entries - (one of each for filter/forward, static/dynamic combinations). - Remove FIB entries. - Test case count: 7
- **VxLAN** - CRD for VxLAN tunnels. - Create VxLAN interface. - Disable VxLAN interface. - Re-create a disabled VxLAN interface. - Test case count: 6
- **VxLAN-GPE** - CRD for VxLAN GPE tunnels. - Create VxLAN GPE interface. - Disable VxLAN interface. - Re-create a disabled VxLAN interface. - Test case count: 7
- **Vhost-user** - CRUD for Vhost-user interfaces. - Create, modify and delete Vhost-user interface, as client and server. - Test case count: 8
- **TAP** - CRUD for Tap interface management. - Create, modify and delete TAP interface. - Test case count: 3
- **VLAN** - CRUD for VLAN sub-interface management. - Create VLAN sub-interface over a physical interface. - Toggle interface state separately for super-interface and sub-interface. - Configure IP address and bridge domain assignment on sub-interface. - Configure VLAN tag rewrite on sub-interface. - Test case count: 24
- **ACL** - CRD for low-level classifiers: table and session management, - interface assignment. - Configure up to 2 classify tables. - Configure up to 2 classify sessions on one table. - Assign classify session to a physical interface. - Remove tables, sessions, interface assignments. - Test case count: 9
- **PBB** - CRD for provider backbone bridge sub-interface. - Configure, modify and remove a PBB sub-interface over a physical interface. - Test case count: 8
- **NSH_SFC** - CRD for NSH maps and entries, using NSH_SFC plugin. - Configure up to 2 NSH entries. - Configure up to 2 NSH maps. - Modify and delete NSH maps and entries. - Test case count: 8
- **LISP** - CRD for Lisp: mapping, locator set, adjacency, map resolver. - Toggle Lisp feature status. - Configure and delete Lisp mapping as local and remote. - Configure and delete Lisp adjacency mapping - Configure and delete Lisp map resolver, proxy ITR. - Test case count: 18
- **LISP GPE** - CRUD for LISP GPE mappings. - Toggle Lisp GPE feature status. - Configure Lisp GPE mappings. - Traffic test verifying encapsulation. - Test case count: 12
- **NAT** - CRD for NAT entries, interface assignment. - Configure and delete up to two NAT entries. - Assign NAT entries to a physical interface. - Test case count: 6
- **Port mirroring** - CRD for SPAN port mirroring, interface assignment. - Configure SPAN port mirroring on a physical interface, mirroring - up to 2 interfaces. - Remove SPAN configuration from interfaces. - Test case count: 14
- **ACL-PLUGIN** - CRD for high-level classifier - MAC + IP address classification. - IPv4, IPv6 address classification. - TCP, UDP, ICMP, ICMPv6 protocol/next-header classification. - port number classification. - ICMP, ICMPv6 code and type classification. - Test case count: 15
- **ProxyARP** - CRD for proxyARP feature. - Configure proxyARP. - Assign to interface. - Test case count: 3
- **ProxyND6** - CRD for Neighbor Discovery Proxy. - Configure ProxyND6 feature on interface. - Test case count: 4

- **DHCP Relay** - CRD for DHCP relay feature. - Configure DHCP Relays. - IPv4 and IPv6 variants. - Test case count: 4
- **SLAAC** - CRD for Stateless Address AutoConfiguration. - Configure SLAAC feature on interfaces. - Test case count: 7
- **Routing** - CRD for routing. - Configure single-hop route. - Configure multi-hop routes. - Configure blackhole route. - IPv4 and IPv6 variants. - Test case count: 6
- **Policer** - CRD for traffic policing feature. - Configure Policing rules. - Assign to interface. - Test case count: 6
- **Border Gateway Protocol** - CRUD and functional tests for BGP. - Configure peers and routes - Check interactions with another BGP peer. - Test case count: 13
- **Honeycomb Infractructure** - configuration persistence, - Netconf notifications for interface events, - Netconf negative tests aimed at specific issues - Netconf/Restconf northbound over IPv6 - Test case count: 12

Total 219 Honeycomb functional tests in the CSIT rls1801_2.

Operational data in Honeycomb should mirror configuration data at all times. Because of this, test cases follow this general pattern:

1. read operational data of the feature using restconf.
2. read status of the feature using VPP API dump.
3. modify configuration of the feature using restconf.
4. verify changes to operational data using restconf.
5. verify changes using VPP API dump, OR
6. send a packet to VPP node and observe behaviour to verify configuration.

Test cases involving network interfaces utilize the first two interfaces on the DUT node.

4.1.3 Functional Tests Naming

CSIT rls1801_2 introduced a common structured naming convention for all performance and functional tests. This change was driven by substantially growing number and type of CSIT test cases. Firstly, the original practice did not always follow any strict naming convention. Secondly test names did not always clearly capture tested packet encapsulations, and the actual type or content of the tests. Thirdly HW configurations in terms of NICs, ports and their locality were not captured either. These were but few reasons that drove the decision to change and define a new more complete and stricter test naming convention, and to apply this to all existing and new test cases.

The new naming should be intuitive for majority of the tests. The complete description of CSIT test naming convention is provided on CSIT test naming page[98].

Here few illustrative examples of the new naming usage for functional test suites:

1. **Physical port to physical port - a.k.a. NIC-to-NIC, Phy-to-Phy, P2P**
   - *eth2p-ethip4-ip4base-func.robot* => 2 ports of Ethernet, IPv4 baseline routed forwarding, functional tests.
2. **Physical port to VM (or VM chain) to physical port - a.k.a. NIC2VM2NIC, P2V2P, NIC2VMchain2NIC, P2V2V2P**
   - *eth2p-ethip4vxlan-l2bdbasemaclrn-eth-2vhost-1vm-func.robot* => 2 ports of Ethernet, IPv4 VXLAN Ethernet, L2 bridge-domain switching to/from two vhost interfaces and one VM, functional tests.

[98] https://wiki.fd.io/view/CSIT/csit-test-naming

4.2 CSIT Release Notes

4.2.1 Changes in CSIT rls1801_2

1. Improved test coverage for the following features:
   - BGP

4.2.2 Known Issues

Here is the list of known issues in CSIT rls1801_2 for Honeycomb functional tests in VIRL:

#	Issue	Jira ID	Description
1	IPv6 BGP route configuration	HONEYCOMB-403	Configuring Ipv6 route results in missing writer for IPv6Route and IPv6NextHop exception.
2	IP address subnet validation	VPP-649	When configuring two IP addresses from the same subnet on an interface, VPP refuses the configuration but returns code 200:OK. This can cause desync between Honeycomb's config and operational data.
3	VxLAN GPE configuration crashes VPP	VPP-875	Specific VxLAN GPE configurations cause VPP to crash and restart.
4	Operational data for IPv6 special routes	HC2VPP-254	Special hop routes are misidentified as regular routes in operational data.
5	LISP PITR feature configuration	HC2VPP-263	Locator set reference in operational data is incorrect.

4.3 Test Environment

CSIT functional tests are currently executed in FD.IO VIRL testbed. See VPP Functional Tests Environment for more details.

4.4 Documentation

CSIT Honeycomb Functional Tests Documentation[99] contains detailed functional description and input parameters for each test case.

[99] https://docs.fd.io/csit/rls1801_2/doc/tests.vpp.func.honeycomb.html

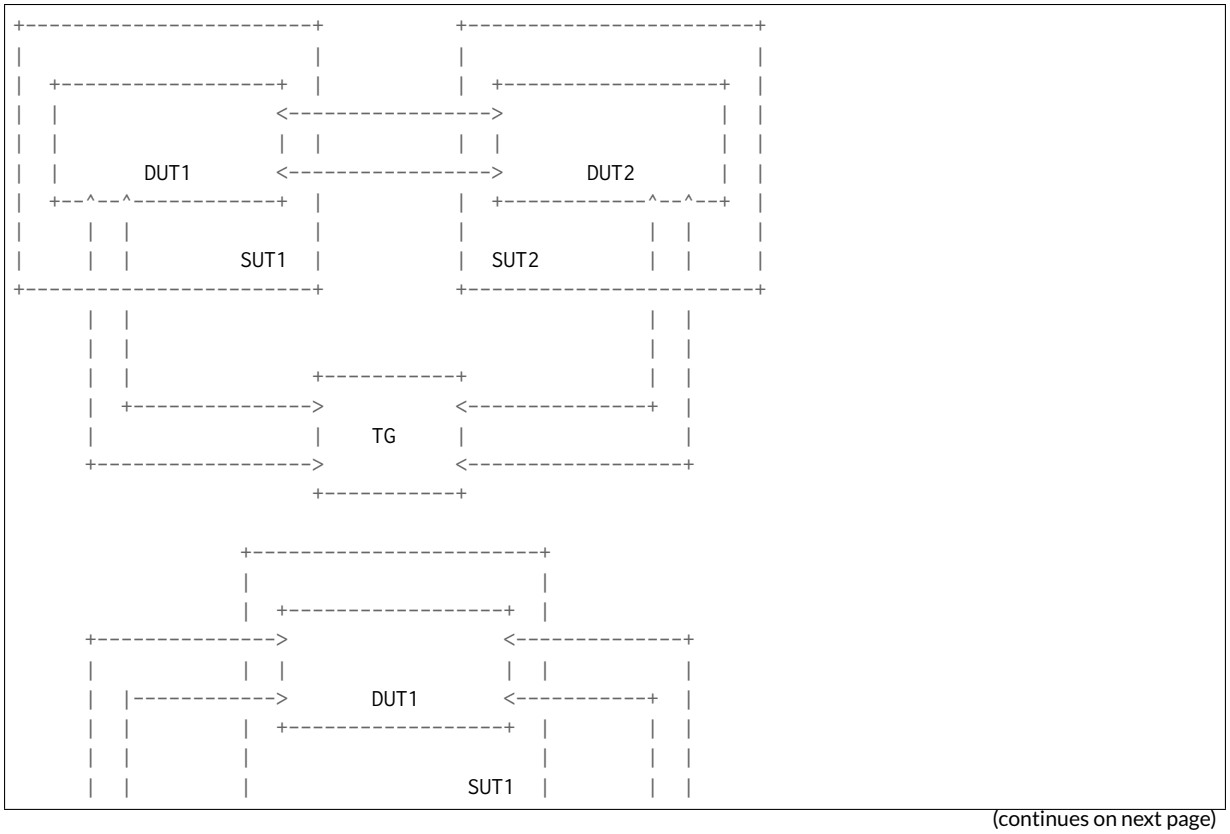
CHAPTER 5

NSH_SFC Functional Tests

5.1 Overview

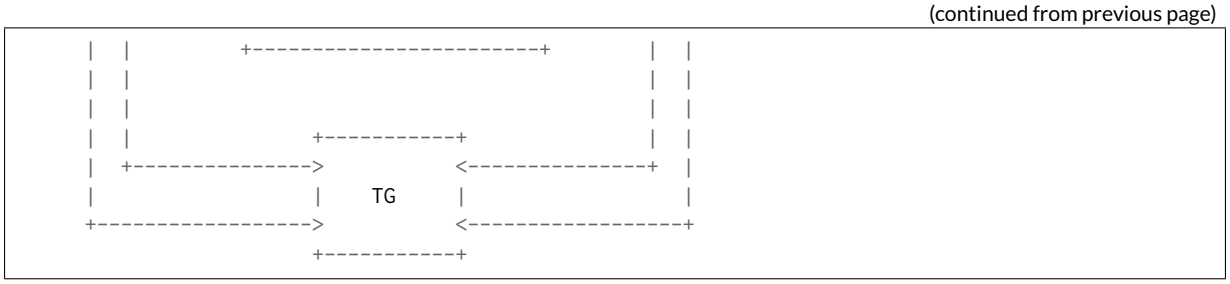
5.1.1 Tested Virtual Topologies

CSIT NSH_SFC functional tests are executed on virtualized topologies created using VIRL simulation platform contributed by Cisco. VIRL runs on physical baremetal servers hosted by LF FD.io project. Majority of the tests are executed in the three node logical test topology - Traffic Generator (TG) node and two Systems Under Test (SUT) nodes connected in a loop. Some tests use two node logical test topology - TG node and SUT1 node. Both logical test topologies are shown in the figures below.:



(continues on next page)

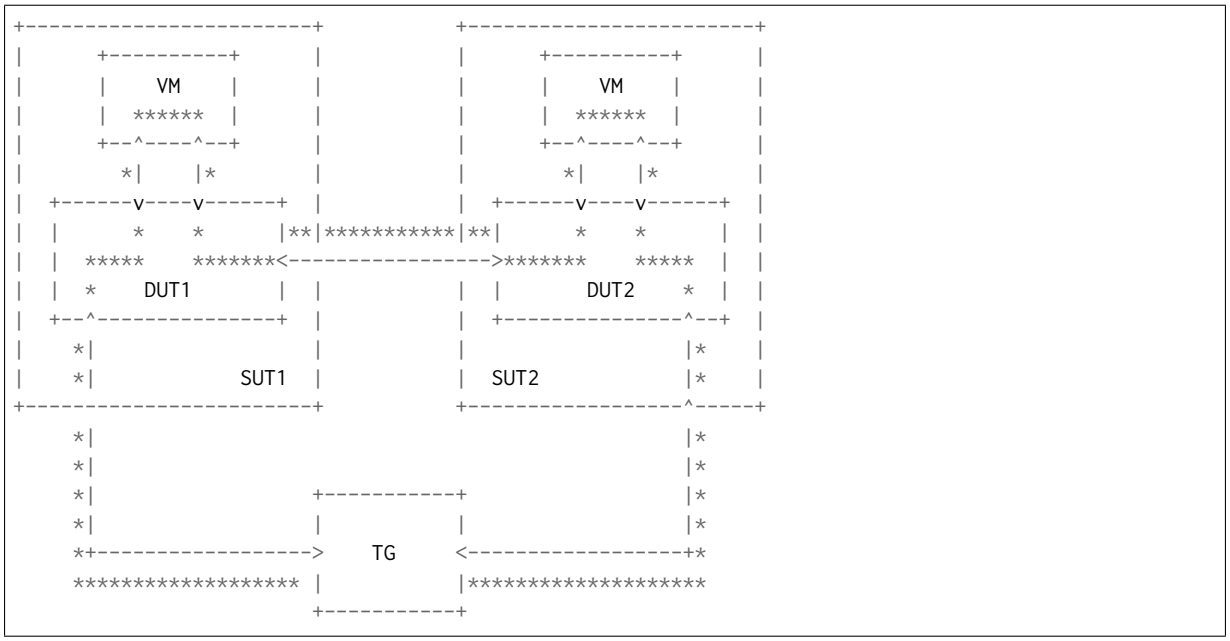
(continued from previous page)



SUT1 and SUT2 are two VMs (Ubuntu or Centos, depending on the test suite), TG is a Traffic Generator (TG, another Ubuntu VM). SUTs run VPP SW application in Linux user-mode as a Device Under Test (DUT) within the VM. TG runs Scapy SW application as a packet Traffic Generator. Logical connectivity between SUTs and to TG is provided using virtual NICs using VMs' virtio driver.

Virtual testbeds are created on-demand whenever a verification job is started (e.g. triggered by the gerrit patch submission) and destroyed upon completion of all functional tests. Each node is a Virtual Machine and each connection that is drawn on the diagram is available for use in any test case. During the test execution, all nodes are reachable thru the Management network connected to every node via dedicated virtual NICs and virtual links (not shown above for clarity).

For the test cases that require DUT (VPP) to communicate with VM over the vhost-user interfaces, a nested VM is created on SUT1 and/or SUT2 for the duration of these particular test cases only. DUT (VPP) test topology with VM is shown in the figure below including the applicable packet flow thru the VM (marked in the figure with ***).:



5.1.2 NSH_SFC Functional Tests Coverage

Following NSH_SFC functional test areas are covered in the CSIT rls1801_2 with results listed in this report:

- **NSH SFC Classifier** - TG sends some TCP packets to test NSH SFC Classifier functional. DUT1 will receive these packets from one NIC and loopback the VXLAN-GPE-NSH encapsulated packets to the TG from other NIC. - Test case count: 7
- **NSH SFC Proxy Inbound** - TG sends some VXLAN-GPE-NSH encapsulated packets to test NSH SFC Proxy Inbound functional. DUT1 will receive these packets from one NIC and loopback the VXLAN encapsulated packets to the TG from other NIC. - Test case count: 6

- **NSH SFC Proxy Outbound** - TG sends some VXLAN encapsulated packets to test NSH SFC Proxy Outbound functional. DUT1 will receive these packets from one NIC and loopback the VXLAN-GPE-NSH encapsulated packets to the TG from other NIC. - Test case count: 6
- **NSH SFC Service Function Forward** - TG sends some VXLAN-GPE-NSH encapsulated packets to test NSH SFC Service Function Forward functional. DUT1 will receive these packets from one NIC and swap the VXLAN-GPE-NSH header, after that DUT1 loopback the VXLAN-GPE-NSH encapsulated packtes to the TG from other NIC. - Test case count: 6

Total 25 NSH SFC functional tests in the CSIT rls1801_2.

5.2 CSIT Release Notes

5.2.1 Changes in CSIT rls1801_2

1. Added NSH SFC functional tests
   - NSH SFC Classifier.
   - NSH SFC Proxy Inbound.
   - NSH SFC Proxy Outbound.
   - NSH SFC SFF.

5.2.2 Known Issues

Here is the list of known issues in CSIT rls1801_2 for NSH_SFC functional tests in VIRL:

#	Issue	Jira ID	Description
1	None	None	None

5.3 Test Environment

CSIT functional tests are currently executed in FD.IO VIRL testbed. The physical VIRL testbed infrastructure consists of three VIRL hosts:

- All hosts are Cisco UCS C240-M4 (2x Intel(R) Xeon(R) CPU E5-2699 v3 @2.30GHz, 18c, 512GB RAM)
- tb4-virl1:
  - Status: Production
  - OS: Ubuntu 16.04.2
  - STD server version 0.10.32.16
  - UWM server version 0.10.32.16
- tb4-virl2:
  - Status: Production
  - OS: Ubuntu 16.04.2
  - STD server version 0.10.32.16
  - UWM server version 0.10.32.16
- tb4-virl3:

- Status: Testing
- OS: Ubuntu 16.04.2
- STD server version 0.10.32.19
- UWM server version 0.10.32.19

Whenever a patch is submitted to gerrit for review, parallel VIRL simulations are started to reduce the time of execution of all functional tests. The number of parallel VIRL simulations is equal to number of test groups defined by TEST_GROUPS variable in `csit/bootstrap.sh` file. The VIRL host to run VIRL simulation is selected based on least load algorithm per VIRL simulation.

Every VIRL simulation uses the same three-node - Traffic Generator (TG node) and two Systems Under Test (SUT1 and SUT2) - "double-ring" topology. The appropriate pre-built VPP packages built by Jenkins for the patch under review are then installed on the two SUTs, along with their `/etc/vpp/startup.conf` file, in all VIRL simulations.

5.3.1 SUT Configuration - VIRL Guest VM

Configurations of the SUT VMs is defined in VIRL topologies directory[100]

- List of SUT VM interfaces::

```
<interface id="0" name="GigabitEthernet0/4/0"/>
<interface id="1" name="GigabitEthernet0/5/0"/>
<interface id="2" name="GigabitEthernet0/6/0"/>
<interface id="3" name="GigabitEthernet0/7/0"/>
```

- Number of 2MB hugepages: 1024
- Maximum number of memory map areas: 20000
- Kernel Shared Memory Max: 2147483648 (vm.nr_hugepages * 2 * 1024 * 1024)

5.3.2 SUT Configuration - VIRL Guest OS Linux

In CSIT terminology, the VM operating system for both SUTs that VPP-18.01.2 release has been tested with, is the following:

1. **Ubuntu VIRL image**

   This image implies Ubuntu 16.04.1 LTS, current as of yyyy-mm-dd (that is, package versions are those that would have been installed by a **`apt-get update`**, **`apt-get upgrade`** on that day), produced by CSIT disk image build scripts.

   The exact list of installed packages and their versions (including the Linux kernel package version) are included in VIRL images lists[101].

   A replica of this VM image can be built by running the **`build.sh`** script in CSIT repository.

2. **CentOS VIRL image**

   The Centos7.3 image is ready to be used but no tests running on it now. Corresponding Jenkins jobs are under preparation.

   The exact list of installed packages and their versions (including the Linux kernel package version) are included in VIRL images lists[102].

   A replica of this VM image can be built by running the **`build.sh`** script in CSIT repository.

[100] https://git.fd.io/csit/tree/resources/tools/virl/topologies/?h=rls1801_2
[101] https://git.fd.io/csit/tree/resources/tools/disk-image-builder/ubuntu/lists/?h=rls1801_2
[102] https://git.fd.io/csit/tree/resources/tools/disk-image-builder/ubuntu/lists/?h=rls1801_2

3. **Nested VM image**

   In addition to the "main" VM image, tests which require VPP to communicate to a VM over a vhost-user interface, utilize a "nested" VM image.

   This "nested" VM is dynamically created and destroyed as part of a test case, and therefore the "nested" VM image is optimized to be small, lightweight and have a short boot time. The "nested" VM image is not built around any established Linux distribution, but is based on BuildRoot[103], a tool for building embedded Linux systems. Just as for the "main" image, scripts to produce an identical replica of the "nested" image are included in CSIT GIT repository, and the image can be rebuilt using the "build.sh" script at VIRL nested[104].

5.3.3 DUT Configuration - VPP

Every System Under Test runs VPP SW application in Linux user-mode as a Device Under Test (DUT) node.

DUT port configuration

Port configuration of DUTs is defined in topology file that is generated per VIRL simulation based on the definition stored in VIRL topologies directory[105].

Example of DUT nodes configuration::

```
DUT1:
    type: DUT
    host: "10.30.51.157"
    port: 22
    username: cisco
    honeycomb:
      user: admin
      passwd: admin
      port: 8183
      netconf_port: 2831
    priv_key: |
      -----BEGIN RSA PRIVATE KEY-----
      MIIEpgIBAAKCAQEAwUDlTpzSHpwLQotZOFS4AgcPNEWCnP1AB2hWFmvI+8Kah/gb
      v8ruZU9RqhPs56tyKzxbhvNkY4VbH5F1GilHZu3mLqzM4KfghMmaeMEjO1T7BYYd
      vuBfTvIluljfQ2vAlnYrDwn+ClxJk81m0pDgvrLEX4qVVh2sGh7UEkYy5r82DNa2
      4VjzPB1J/c8a9zP8FoZUhYIzF4FLvRMjUADpbMXgJMsGpaZLmz95ap0Eot7vb1Cc
      1LvF97iyBCrtIOSKRKA50ZhLGjMKmOwnYU+cP5718tbproDVi6VJOo7zeuXyetMs
      8YBl9kWblWG9BqP9jctFvsmi5G7hXgq1Y8u+DwIDAQABAoIBAQC/W4E0DHjLMny7
      0bvw2YKzD0Zw3fttdB94tkm4PdZv5MybooPnsAvLaXVV0hEdfVi5kzSWNl/LY/tN
      EP1BgGphc2QgB59/PPxGwFIjDCvUzlsZpynBHe+B/qh5ExNQcVvsIOqWI7DXlXaN
      0i/khOzmJ6HncRRah1spKimYRsaUUDskyg7q3QqMWVaqBbbMvLs/w7ZWd/zoDqCU
      MY/pCI6hkB3QbRo0OdiZLohphBl2ShABTwjvVyyKL5UA4jAEneJrhH5gWVLXnfgD
      p62W5CollKEYblC8mUkPxpP7Qo277zw3xaq+oktIZhc5SUEUd7nJZtNqVAHqkItW
      79VmpKyxAoGBAPfU+kqNPaTSvp+x1n5sn2SgipzDtgi9QqNmC4cjtrQQaaqI57SG
      OHw1jX8i7L2G1WvVtkHg060nlEVo5n65ffFOqeVBezLVJ7ghWI8U+oBiJJyQ4boD
      GJVNsoOSUQ0rtuGd9eVwfDk3ol9aCN0KK53oPfIYli29pyu4l095kg11AoGBAMef
      bPEMBI/2XmCPshLSwhGFl+dW8d+Klluj3CUQ/0vUlvma3dfBOYNsIwAgTP0iIUTg
      8DYE6KBCdPtxAUEI0YAEAKB9ry1tKR2NQEIPfslYytKErtwjAiqSi0heM6+zwEzu
      f54Z4oBhsMSL0jXoOMnu+NZzEc6EUdQeY4O+jhjzAoGBAIogC3dtjMPGKTP7+93u
      UE/XIioI8fWg9fj3sMka4IMu+pVvRCRbAjRH7JrFLkjbUyuMqs3Arnk9K+gbdQt/
      +m95Njtt6WoFXuPCwgbM3GidSmZwYT4454SfDzVBYScEDCNm1FuR+8ov9bFLDtGT
      D4gsngnGJj1MDFXTxZEn4nzZAoGBAKCg4WmpUPaCuXibyB+rZavxwsTNSn2lJ83/
      sYJGBhf/raiV/FLDUcM1vYg5dZnu37RsB/5/vqxOLZGyYd7x+Jo5HkQGPnKgNwhn
      g8BkdZIRF8uEJqxOo0ycdOU7n/2O93swIpKWo5LIiRPuqqzj+uZKnAL7vuVdxfaY
      qVz2daMPAoGBALgaaKa3voU/HO1PYLWIhFrBThyJ+BQSQ8OqrEzC8AnegWFxRAM8
      EqrzZXl7ACUuo1dH0Eipm41j2+BZWlQjiUgq5uj8+yzy+EU1ZRRyJcOKzbDACeuD
```

(continues on next page)

[103] https://buildroot.org/
[104] https://git.fd.io/csit/tree/resources/tools/disk-image-builder/nested/?h=rls1801_2
[105] https://git.fd.io/csit/tree/resources/tools/virl/topologies/?h=rls1801_2

(continued from previous page)

```
    BpWWSXGBI5G4CppeYLjMUHZpJYeX1USULJQd2c4crLJKb76E8gz3Z9kN
    -----END RSA PRIVATE KEY-----

  interfaces:
    port1:
      mac_address: "fa:16:3e:9b:89:52"
      pci_address: "0000:00:04.0"
      link: link1
    port2:
      mac_address: "fa:16:3e:7a:33:60"
      pci_address: "0000:00:05.0"
      link: link4
    port3:
      mac_address: "fa:16:3e:29:b7:ae"
      pci_address: "0000:00:06.0"
      link: link3
    port4:
      mac_address: "fa:16:3e:76:8d:ff"
      pci_address: "0000:00:07.0"
      link: link6
DUT2:
  type: DUT
  host: "10.30.51.156"
  port: 22
  username: cisco
  honeycomb:
    user: admin
    passwd: admin
    port: 8183
    netconf_port: 2831
  priv_key: |
    -----BEGIN RSA PRIVATE KEY-----
    MIIEpgIBAAKCAQEAwUDlTpzSHpwLQotZOFS4AgcPNEWCnP1AB2hWFmvI+8Kah/gb
    v8ruZU9RqhPs56tyKzxbhvNkY4VbH5F1GilHZu3mLqzM4KfghMmaeMEjO1T7BYYd
    vuBfTvIluljfQ2vAlnYrDwn+ClxJk81m0pDgvrLEX4qVVh2sGh7UEkYy5r82DNa2
    4VjzPB1J/c8a9zP8FoZUhYIzF4FLvRMjUADpbMXgJMsGpaZLmz95ap0Eot7vb1Cc
    1LvF97iyBCrtIOSKRKA50ZhLGjMKmOwnYU+cP5718tbproDVi6VJOo7zeuXyetMs
    8YBl9kWblWG9BqP9jctFvsmi5G7hXgq1Y8u+DwIDAQABAoIBAQC/W4E0DHjLMny7
    0bvw2YKzD0Zw3fttdB94tkm4PdZv5MybooPnsAvLaXVV0hEdfVi5kzSWNl/LY/tN
    EP1BgGphc2QgB59/PPxGwFIjDCvUzlsZpynBHe+B/qh5ExNQcVvsIOqWI7DXlXaN
    0i/khOzmJ6HncRRah1spKimYRsaUUDskyg7q3QqMWVaqBbbMvLs/w7ZWd/zoDqCU
    MY/pCI6hkB3QbRo0OdiZLohphBl2ShABTwjvVyyKL5UA4jAEneJrhH5gWVLXnfgD
    p62W5CollKEYblC8mUkPxpP7Qo277zw3xaq+oktIZhc5SUEUd7nJZtNqVAHqkItW
    79VmpKyxAoGBAPfU+kqNPaTSvp+x1n5sn2SgipzDtgi9QqNmC4cjtrQQaaqI57SG
    OHw1jX8i7L2G1WvVtkHg060nlEVo5n65ffFOqeVBezLVJ7ghWI8U+oBiJJyQ4boD
    GJVNsoOSUQ0rtuGd9eVwfDk3ol9aCN0KK53oPfIYli29pyu4l095kg11AoGBAMef
    bPEMBI/2XmCPshLSwhGFl+dW8d+Klluj3CUQ/0vUlvma3dfBOYNsIwAgTP0iIUTg
    8DYE6KBCdPtxAUEI0YAEAKB9ry1tKR2NQEIPfslYytKErtwjAiqSi0heM6+zwEzu
    f54Z4oBhsMSL0jXoOMnu+NZzEc6EUdQeY4O+jhjzAoGBAIogC3dtjMPGKTP7+93u
    UE/XIioI8fWg9fj3sMka4IMu+pVvRCRbAjRH7JrFLkjbUyuMqs3Arnk9K+gbdQt/
    +m95Njtt6WoFXuPCwgbM3GidSmZwYT4454SfDzVBYScEDCNm1FuR+8ov9bFLDtGT
    D4gsngnGJj1MDFXTxZEn4nzZAoGBAKCg4WmpUPaCuXibyB+rZavxwsTNSn2lJ83/
    sYJGBhf/raiV/FLDUcM1vYg5dZnu37RsB/5/vqxOLZGyYd7x+Jo5HkQGPnKgNwhn
    g8BkdZIRF8uEJqxOo0ycdOU7n/2O93swIpKWo5LIiRPuqqzj+uZKnAL7vuVdxfaY
    qVz2daMPAoGBALgaaKa3voU/HO1PYLWIhFrBThyJ+BQSQ8OqrEzC8AnegWFxRAM8
    EqrzZXl7ACUuo1dH0Eipm41j2+BZWlQjiUgq5uj8+yzy+EU1ZRRyJcOKzbDACeuD
    BpWWSXGBI5G4CppeYLjMUHZpJYeX1USULJQd2c4crLJKb76E8gz3Z9kN
    -----END RSA PRIVATE KEY-----

  interfaces:
    port1:
```

(continues on next page)

(continued from previous page)

```
        mac_address: "fa:16:3e:ad:6c:7d"
        pci_address: "0000:00:04.0"
        link: link2
      port2:
        mac_address: "fa:16:3e:94:a4:99"
        pci_address: "0000:00:05.0"
        link: link5
      port3:
        mac_address: "fa:16:3e:75:92:da"
        pci_address: "0000:00:06.0"
        link: link3
      port4:
        mac_address: "fa:16:3e:2c:b1:2a"
        pci_address: "0000:00:07.0"
        link: link6
```

VPP Version

VPP-18.01.2 release

VPP Installed Packages

```
$ dpkg -l vpp\*
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name           Version       Architecture Description
+++-==============-=============-============-=================================================
ii  vpp            17.07-release amd64        Vector Packet Processing--executables
ii  vpp-dbg        17.07-release amd64        Vector Packet Processing--debug symbols
ii  vpp-dev        17.07-release amd64        Vector Packet Processing--development support
ii  vpp-dpdk-dev   17.07-release amd64        Vector Packet Processing--development support
ii  vpp-dpdk-dkms  17.07-release amd64        DPDK 2.1 igb_uio_driver
ii  vpp-lib        17.07-release amd64        Vector Packet Processing--runtime libraries
ii  vpp-plugins    17.07-release amd64        Vector Packet Processing--runtime plugins
```

VPP Startup Configuration

VPP startup configuration is common for all test cases.

```
$ cat /etc/vpp/startup.conf
unix {
  nodaemon
  log /tmp/vpp.log
  full-coredump
}

api-trace {
  on
}

api-segment {
  gid vpp
}

cpu {
    ## In the VPP there is one main thread and optionally the user can create worker(s)
    ## The main thread and worker thread(s) can be pinned to CPU core(s) manually or automatically

    ## Manual pinning of thread(s) to CPU core(s)

    ## Set logical CPU core where main thread runs
```

(continues on next page)

(continued from previous page)

```
    # main-core 1

    ## Set logical CPU core(s) where worker threads are running
    # corelist-workers 2-3,18-19

    ## Automatic pinning of thread(s) to CPU core(s)

    ## Sets number of CPU core(s) to be skipped (1 ... N-1)
    ## Skipped CPU core(s) are not used for pinning main thread and working thread(s).
    ## The main thread is automatically pinned to the first available CPU core and worker(s)
    ## are pinned to next free CPU core(s) after core assigned to main thread
    # skip-cores 4

    ## Specify a number of workers to be created
    ## Workers are pinned to N consecutive CPU cores while skipping "skip-cores" CPU core(s)
    ## and main thread's CPU core
    # workers 2

    ## Set scheduling policy and priority of main and worker threads

    ## Scheduling policy options are: other (SCHED_OTHER), batch (SCHED_BATCH)
    ## idle (SCHED_IDLE), fifo (SCHED_FIFO), rr (SCHED_RR)
    # scheduler-policy fifo

    ## Scheduling priority is used only for "real-time policies (fifo and rr),
    ## and has to be in the range of priorities supported for a particular policy
    # scheduler-priority 50
}

dpdk {
    ## Change default settings for all intefaces
    # dev default {
        ## Number of receive queues, enables RSS
        ## Default is 1
        # num-rx-queues 3

        ## Number of transmit queues, Default is equal
        ## to number of worker threads or 1 if no workers treads
        # num-tx-queues 3

        ## Number of descriptors in transmit and receive rings
        ## increasing or reducing number can impact performance
        ## Default is 1024 for both rx and tx
        # num-rx-desc 512
        # num-tx-desc 512

        ## VLAN strip offload mode for interface
        ## Default is off
        # vlan-strip-offload on
    # }

    ## Whitelist specific interface by specifying PCI address
    # dev 0000:02:00.0

    ## Whitelist specific interface by specifying PCI address and in
    ## addition specify custom parameters for this interface
    # dev 0000:02:00.1 {
    #       num-rx-queues 2
    # }

    ## Change UIO driver used by VPP, Options are: uio_pci_generic, vfio-pci
```

(continues on next page)

(continued from previous page)

```
    ## and igb_uio (default)
    # uio-driver uio_pci_generic

    ## Disable mutli-segment buffers, improves performance but
    ## disables Jumbo MTU support
    # no-multi-seg

    ## Increase number of buffers allocated, needed only in scenarios with
    ## large number of interfaces and worker threads. Value is per CPU socket.
    ## Default is 32768
    # num-mbufs 128000

    ## Change hugepages allocation per-socket, needed only if there is need for
    ## larger number of mbufs. Default is 256M on each detected CPU socket
    # socket-mem 2048,2048
}
```

5.3.4 TG Configuration

Traffic Generator node is VM running the same OS Linux as SUTs. Ports of this VM are used as source (Tx) and destination (Rx) ports for the traffic.

Traffic scripts of test cases are executed on this VM.

TG VM configuration

Configuration of the TG VMs is defined in VIRL topologies directory[106].

/csit/resources/tools/virl/topologies/double-ring-nested.xenial.virl

- List of TG VM interfaces::

```
<interface id="0" name="eth1"/>
<interface id="1" name="eth2"/>
<interface id="2" name="eth3"/>
<interface id="3" name="eth4"/>
<interface id="4" name="eth5"/>
<interface id="5" name="eth6"/>
```

TG node port configuration

Port configuration of TG is defined in topology file that is generated per VIRL simulation based on the definition stored in VIRL topologies directory[107].

Example of TG node configuration::

```
TG:
    type: TG
    host: "10.30.51.155"
    port: 22
    username: cisco
    priv_key: |
      -----BEGIN RSA PRIVATE KEY-----
      MIIEpgIBAAKCAQEAwUDlTpzSHpwLQotZOFS4AgcPNEWCnP1AB2hWFmvI+8Kah/gb
      v8ruZU9RqhPs56tyKzxbhvNkY4VbH5F1GilHZu3mLqzM4KfghMmaeMEjO1T7BYYd
      vuBfTvIluljfQ2vAlnYrDwn+ClxJk81m0pDgvrLEX4qVVh2sGh7UEkYy5r82DNa2
      4VjzPB1J/c8a9zP8FoZUhYIzF4FLvRMjUADpbMXgJMsGpaZLmz95ap0Eot7vb1Cc
      1LvF97iyBCrtIOSKRKA50ZhLGjMKmOwnYU+cP5718tbproDVi6VJOo7zeuXyetMs
      8YBl9kWblWG9BqP9jctFvsmi5G7hXgq1Y8u+DwIDAQABAoIBAQC/W4E0DHjLMny7
```

(continues on next page)

[106] https://git.fd.io/csit/tree/resources/tools/virl/topologies/?h=rls1801_2
[107] https://git.fd.io/csit/tree/resources/tools/virl/topologies/?h=rls1801_2

(continued from previous page)

```
    0bvw2YKzD0Zw3fttdB94tkm4PdZv5MybooPnsAvLaXVV0hEdfVi5kzSWNl/LY/tN
    EP1BgGphc2QgB59/PPxGwFIjDCvUzlsZpynBHe+B/qh5ExNQcVvsIOqWI7DXlXaN
    0i/khOzmJ6HncRRah1spKimYRsaUUDskyg7q3QqMWVaqBbbMvLs/w7ZWd/zoDqCU
    MY/pCI6hkB3QbRo0OdiZLohphBl2ShABTwjvVyyKL5UA4jAEneJrhH5gWVLXnfgD
    p62W5CollKEYblC8mUkPxpP7Qo277zw3xaq+oktIZhc5SUEUd7nJZtNqVAHqkItW
    79VmpKyxAoGBAPfU+kqNPaTSvp+x1n5sn2SgipzDtgi9QqNmC4cjtrQQaaqI57SG
    OHw1jX8i7L2G1WvVtkHg060nlEVo5n65ffFOqeVBezLVJ7ghWI8U+oBiJJyQ4boD
    GJVNsoOSUQ0rtuGd9eVwfDk3ol9aCN0KK53oPfIYli29pyu4l095kg11AoGBAMef
    bPEMBI/2XmCPshLSwhGFl+dW8d+Klluj3CUQ/0vUlvma3dfBOYNsIwAgTP0iIUTg
    8DYE6KBCdPtxAUEI0YAEAKB9ry1tKR2NQEIPfslYytKErtwjAiqSi0heM6+zwEzu
    f54Z4oBhsMSL0jXoOMnu+NZzEc6EUdQeY4O+jhjzAoGBAIogC3dtjMPGKTP7+93u
    UE/XIioI8fWg9fj3sMka4IMu+pVvRCRbAjRH7JrFLkjbUyuMqs3Arnk9K+gbdQt/
    +m95Njtt6WoFXuPCwgbM3GidSmZwYT4454SfDzVBYScEDCNm1FuR+8ov9bFLDtGT
    D4gsngnGJj1MDFXTxZEn4nzZAoGBAKCg4WmpUPaCuXibyB+rZavxwsTNSn2lJ83/
    sYJGBhf/raiV/FLDUcM1vYg5dZnu37RsB/5/vqxOLZGyYd7x+Jo5HkQGPnKgNwhn
    g8BkdZIRF8uEJqxOo0ycdOU7n/2O93swIpKWo5LIiRPuqqzj+uZKnAL7vuVdxfaY
    qVz2daMPAoGBALgaaKa3voU/HO1PYLWIhFrBThyJ+BQSQ8OqrEzC8AnegWFxRAM8
    EqrzZXl7ACUuo1dH0Eipm41j2+BZWlQjiUgq5uj8+yzy+EU1ZRRyJcOKzbDACeuD
    BpWWSXGBI5G4CppeYLjMUHZpJYeX1USULJQd2c4crLJKb76E8gz3Z9kN
    -----END RSA PRIVATE KEY-----

  interfaces:
    port3:
      mac_address: "fa:16:3e:b9:e1:27"
      pci_address: "0000:00:06.0"
      link: link1
      driver: virtio-pci
    port4:
      mac_address: "fa:16:3e:e9:c8:68"
      pci_address: "0000:00:07.0"
      link: link4
      driver: virtio-pci
    port5:
      mac_address: "fa:16:3e:e8:d3:47"
      pci_address: "0000:00:08.0"
      link: link2
      driver: virtio-pci
    port6:
      mac_address: "fa:16:3e:cf:ca:58"
      pci_address: "0000:00:09.0"
      link: link5
      driver: virtio-pci
```

Traffic generator

Functional tests utilize Scapy as a traffic generator. There was used Scapy v2.3.1 for VPP-18.01.2 release tests.

5.4 Documentation

CSIT NSH_SFC Functional Tests Documentation[108] contains detailed functional description and input parameters for each test case.

[108] https://docs.fd.io/csit/rls1801_2/doc/tests.nsh_sfc.func.html

CHAPTER 6

CSIT Framework Documentation

6.1 CSIT Design

FD.io CSIT system design needs to meet continuously expanding requirements of FD.io projects including VPP, related sub-systems (e.g. plugin applications, DPDK drivers) and FD.io applications (e.g. DPDK applications), as well as growing number of compute platforms running those applications. With CSIT project scope and charter including both FD.io continuous testing AND performance trending/comparisons, those evolving requirements further amplify the need for CSIT framework modularity, flexibility and usability.

6.1.1 Design Hierarchy

CSIT follows a hierarchical system design with SUTs and DUTs at the bottom level of the hierarchy, presentation level at the top level and a number of functional layers in-between. The current CSIT system design including CSIT framework is depicted in the figure below.

csit_design_picture.pdf

A brief bottom-up description is provided here:

1. SUTs, DUTs, TGs
   - SUTs - Systems Under Test;
   - DUTs - Devices Under Test;
   - TGs - Traffic Generators;
2. Level-1 libraries - Robot and Python
   - Lowest level CSIT libraries abstracting underlying test environment, SUT, DUT and TG specifics;
   - Used commonly across multiple L2 KWs;
   - Performance and functional tests:
     - L1 KWs (KeyWords) are implemented as RF libraries and Python libraries;
   - Performance TG L1 KWs:
     - All L1 KWs are implemented as Python libraries:
       * Support for TRex only today;
       * CSIT IXIA drivers in progress;
   - Performance data plane traffic profiles:
     - TG-specific stream profiles provide full control of:
       * Packet definition – layers, MACs, IPs, ports, combinations thereof e.g. IPs and UDP ports;
       * Stream definitions - different streams can run together, delayed, one after each other;
       * Stream profiles are independent of CSIT framework and can be used in any T-rex setup, can be sent anywhere to repeat tests with exactly the same setup;

        * Easily extensible – one can create a new stream profile that meets tests requirements;
        * Same stream profile can be used for different tests with the same traffic needs;
    - Functional data plane traffic scripts:
        - Scapy specific traffic scripts;
3. Level-2 libraries - Robot resource files:
    - Higher level CSIT libraries abstracting required functions for executing tests;
    - L2 KWs are classified into the following functional categories:
        - Configuration, test, verification, state report;
        - Suite setup, suite teardown;
        - Test setup, test teardown;
4. Tests - Robot:
    - Test suites with test cases;
    - Functional tests using VIRL environment:
        - VPP;
        - Honeycomb;
        - NSH_SFC;
    - Performance tests using physical testbed environment:
        - VPP;
        - DPDK-Testpmd;
        - DPDK-L3Fwd;
        - Honeycomb;
        - VPP Container K8s orchestrated topologies;
    - Tools:
        - Documentation generator;
        - Report generator;
        - Testbed environment setup ansible playbooks;
        - Operational debugging scripts;

6.1.2 Test Lifecycle Abstraction

A well coded test must follow a disciplined abstraction of the test lifecycles that includes setup, configuration, test and verification. In addition to improve test execution efficiency, the commmon aspects of test setup and configuration shared across multiple test cases should be done only once. Translating these high-level guidelines into the Robot Framework one arrives to definition of a well coded RF tests for FD.io CSIT. Anatomy of Good Tests for CSIT:

1. Suite Setup - Suite startup Configuration common to all Test Cases in suite: uses Configuration KWs, Verification KWs, StateReport KWs;
2. Test Setup - Test startup Configuration common to multiple Test Cases: uses Configuration KWs, StateReport KWs;
3. Test Case - uses L2 KWs with RF Gherkin style:
    - prefixed with {Given} - Verification of Test setup, reading state: uses Configuration KWs, Verification KWs, StateReport KWs;

- prefixed with {When} - Test execution: Configuration KWs, Test KWs;
- prefixed with {Then} - Verification of Test execution, reading state: uses Verification KWs, StateReport KWs;

4. Test Teardown - post Test teardown with Configuration cleanup and Verification common to multiple Test Cases - uses: Configuration KWs, Verification KWs, StateReport KWs;
5. Suite Teardown - Suite post-test Configuration cleanup: uses Configuration KWs, Verification KWs, StateReport KWs;

6.1.3 RF Keywords Functional Classification

CSIT RF KWs are classified into the functional categories matching the test lifecycle events described earlier. All CSIT RF L2 and L1 KWs have been grouped into the following functional categories:

1. Configuration;
2. Test;
3. Verification;
4. StateReport;
5. SuiteSetup;
6. TestSetup;
7. SuiteTeardown;
8. TestTeardown;

6.1.4 RF Keywords Naming Guidelines

Readability counts: "..code is read much more often than it is written." Hence following a good and consistent grammar practice is important when writing RF KeyWords and Tests. All CSIT test cases are coded using Gherkin style and include only L2 KWs references. L2 KWs are coded using simple style and include L2 KWs, L1 KWs, and L1 python references. To improve readability, the proposal is to use the same grammar for both RF KW styles, and to formalize the grammar of English sentences used for naming the RF KWs. RF KWs names are short sentences expressing functional description of the command. They must follow English sentence grammar in one of the following forms:

1. **Imperative** - verb-object(s): *"Do something"*, verb in base form.
2. **Declarative** - subject–verb–object(s): *"Subject does something"*, verb in a third-person singular present tense form.
3. **Affirmative** - modal_verb-verb-object(s): *"Subject should be something"*, *"Object should exist"*, verb in base form.
4. **Negative** - modal_verb-Not-verb-object(s): *"Subject should not be something"*, *"Object should not exist"*, verb in base form.

Passive form MUST NOT be used. However a usage of past participle as an adjective is okay. See usage examples provided in the Coding guidelines section below. Following sections list applicability of the above grammar forms to different RF KW categories. Usage examples are provided, both good and bad.

6.1.5 Coding guidelines

Coding guidelines can be found on Design optimizations wiki page[109].

[109] https://wiki.fd.io/view/CSIT/Design_Optimizations

6.2 CSIT Test Naming

6.2.1 Background

CSIT rls1801_2 follows a common structured naming convention for all performance and system functional tests, introduced in CSIT rls1710.

The naming should be intuitive for majority of the tests. Complete description of CSIT test naming convention is provided on CSIT test naming wiki page[110]. Below few illustrative examples of the naming usage for test suites across CSIT performance, functional and Honeycomb management test areas.

6.2.2 Naming Convention

The CSIT approach is to use tree naming convention and to encode following testing information into test suite and test case names:

1. packet network port configuration
    - port type, physical or virtual;
    - number of ports;
    - NIC model, if applicable;
    - port-NIC locality, if applicable;
2. packet encapsulations;
3. VPP packet processing
    - packet forwarding mode;
    - packet processing function(s);
4. packet forwarding path
    - if present, network functions (processes, containers, VMs) and their topology within the computer;
5. main measured variable, type of test.

Proposed convention is to encode ports and NICs on the left (underlay), followed by outer-most frame header, then other stacked headers up to the header processed by vSwitch-VPP, then VPP forwarding function, then encap on vhost interface, number of vhost interfaces, number of VMs. If chained VMs present, they get added on the right. Test topology is expected to be symmetric, in other words packets enter and leave SUT through ports specified on the left of the test name. Here some examples to illustrate the convention followed by the complete legend, and tables mapping the new test filenames to old ones.

6.2.3 Naming Examples

CSIT test suite naming examples (filename.robot) for common tested VPP topologies:

1. **Physical port to physical port - a.k.a. NIC-to-NIC, Phy-to-Phy, P2P**
    - *PortNICConfig-WireEncapsulation-PacketForwardingFunction-PacketProcessingFunction1-...-PacketProcessingFunctionN-TestType*
    - *10ge2p1x520-dot1q-l2bdbasemaclrn-ndrdisc.robot* => 2 ports of 10GE on Intel x520 NIC, dot1q tagged Ethernet, L2 bridge-domain baseline switching with MAC learning, NDR throughput discovery.

[110] https://wiki.fd.io/view/CSIT/csit-test-naming

- *10ge2p1x520-ethip4vxlan-l2bdbasemaclrn-ndrchk.robot* => 2 ports of 10GE on Intel x520 NIC, IPv4 VXLAN Ethernet, L2 bridge-domain baseline switching with MAC learning, NDR throughput discovery.
- *10ge2p1x520-ethip4-ip4base-ndrdisc.robot* => 2 ports of 10GE on Intel x520 NIC, IPv4 baseline routed forwarding, NDR throughput discovery.
- *10ge2p1x520-ethip6-ip6scale200k-ndrdisc.robot* => 2 ports of 10GE on Intel x520 NIC, IPv6 scaled up routed forwarding, NDR throughput discovery.
- *10ge2p1x520-ethip4-ip4base-iacldstbase-ndrdisc.robot* => 2 ports of 10GE on Intel x520 NIC, IPv4 baseline routed forwarding, ingress Access Control Lists baseline matching on destination, NDR throughput discovery.
- *40ge2p1vic1385-ethip4-ip4base-ndrdisc.robot* => 2 ports of 40GE on Cisco vic1385 NIC, IPv4 baseline routed forwarding, NDR throughput discovery.
- *eth2p-ethip4-ip4base-func.robot* => 2 ports of Ethernet, IPv4 baseline routed forwarding, functional tests.

2. **Physical port to VM (or VM chain) to physical port - a.k.a. NIC2VM2NIC, P2V2P, NIC2VMchain2NIC, P2V2V2P**
   - *PortNICConfig-WireEncapsulation-PacketForwardingFunction- PacketProcessingFunction1-…-PacketProcessingFunctionN-VirtEncapsulation- VirtPortConfig-VMconfig-TestType*
   - *10ge2p1x520-dot1q-l2bdbasemaclrn-eth-2vhost-1vm-ndrdisc.robot* => 2 ports of 10GE on Intel x520 NIC, dot1q tagged Ethernet, L2 bridge-domain switching to/from two vhost interfaces and one VM, NDR throughput discovery.
   - *10ge2p1x520-ethip4vxlan-l2bdbasemaclrn-eth-2vhost-1vm-ndrdisc.robot* => 2 ports of 10GE on Intel x520 NIC, IPv4 VXLAN Ethernet, L2 bridge-domain switching to/from two vhost interfaces and one VM, NDR throughput discovery.
   - *10ge2p1x520-ethip4vxlan-l2bdbasemaclrn-eth-4vhost-2vm-ndrdisc.robot* => 2 ports of 10GE on Intel x520 NIC, IPv4 VXLAN Ethernet, L2 bridge-domain switching to/from four vhost interfaces and two VMs, NDR throughput discovery.
   - *eth2p-ethip4vxlan-l2bdbasemaclrn-eth-2vhost-1vm-func.robot* => 2 ports of Ethernet, IPv4 VXLAN Ethernet, L2 bridge-domain switching to/from two vhost interfaces and one VM, functional tests.
3. **API CRUD tests - Create (Write), Read (Retrieve), Update (Modify), Delete (Destroy) operations for configuration and operational data**
   - *ManagementTestKeyword-ManagementOperation-ManagedFunction1-…- ManagedFunctionN-ManagementAPI1-ManagementAPIN-TestType*
   - *mgmt-cfg-lisp-apivat-func* => configuration of LISP with VAT API calls, functional tests.
   - *mgmt-cfg-l2bd-apihc-apivat-func* => configuration of L2 Bridge-Domain with Honeycomb API and VAT API calls, functional tests.
   - *mgmt-oper-int-apihcnc-func* => reading status and operational data of interface with Honeycomb NetConf API calls, functional tests.
   - *mgmt-cfg-int-tap-apihcnc-func* => configuration of tap interfaces with Honeycomb NetConf API calls, functional tests.
   - *mgmt-notif-int-subint-apihcnc-func* => notifications of interface and sub-interface events with Honeycomb NetConf Notifications, functional tests.

For complete description of CSIT test naming convention please refer to CSIT test naming wiki page[111].

[111] https://wiki.fd.io/view/CSIT/csit-test-naming

6.3 Presentation and Analytics Layer

6.3.1 Overview

The presentation and analytics layer (PAL) is the fourth layer of CSIT hierarchy. The model of presentation and analytics layer consists of four sub-layers, bottom up:

- sL1 - Data - input data to be processed:
  - Static content - .rst text files, .svg static figures, and other files stored in the CSIT git repository.
  - Data to process - .xml files generated by Jenkins jobs executing tests, stored as robot results files (output.xml).
  - Specification - .yaml file with the models of report elements (tables, plots, layout, …) generated by this tool. There is also the configuration of the tool and the specification of input data (jobs and builds).
- sL2 - Data processing
  - The data are read from the specified input files (.xml) and stored as multi-indexed pandas.Series[112].
  - This layer provides also interface to input data and filtering of the input data.
- sL3 - Data presentation - This layer generates the elements specified in the specification file:
  - Tables: .csv files linked to static .rst files.
  - Plots: .html files generated using plot.ly linked to static .rst files.
- sL4 - Report generation - Sphinx generates required formats and versions:
  - formats: html, pdf
  - versions: minimal, full (TODO: define the names and scope of versions)

[illegible], Input data (Pandas.Series)

pal_layers.pdf

[112] https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.html

6.3.2 Data

Report Specification

The report specification file defines which data is used and which outputs are generated. It is human readable and structured. It is easy to add / remove / change items. The specification includes:

- Specification of the environment.
- Configuration of debug mode (optional).
- Specification of input data (jobs, builds, files, ...).
- Specification of the output.
- What and how is generated: - What: plots, tables. - How: specification of all properties and parameters.
- .yaml format.

Structure of the specification file

The specification file is organized as a list of dictionaries distinguished by the type:

```
-
  type: "environment"
-
  type: "configuration"
-
  type: "debug"
-
  type: "static"
-
  type: "input"
-
  type: "output"
-
  type: "table"
-
  type: "plot"
-
  type: "file"
```

Each type represents a section. The sections "environment", "debug", "static", "input" and "output" are listed only once in the specification; "table", "file" and "plot" can be there multiple times.

Sections "debug", "table", "file" and "plot" are optional.

Table(s), files(s) and plot(s) are referred as "elements" in this text. It is possible to define and implement other elements if needed.

Section: Environment

This section has the following parts:

- type: "environment" - says that this is the section "environment".
- configuration - configuration of the PAL.
- paths - paths used by the PAL.
- urls - urls pointing to the data sources.
- make-dirs - a list of the directories to be created by the PAL while preparing the environment.

- remove-dirs - a list of the directories to be removed while cleaning the environment.
- build-dirs - a list of the directories where the results are stored.

The structure of the section "Environment" is as follows (example):

```
-
  type: "environment"
  configuration:
    # Debug mode:
    # - Skip:
    #   - Download of input data files
    # - Do:
    #   - Read data from given zip / xml files
    #   - Set the configuration as it is done in normal mode
    # If the section "type: debug" is missing, CFG[DEBUG] is set to 0.
    CFG[DEBUG]: 0

  paths:
    # Top level directories:
    ## Working directory
    DIR[WORKING]: "_tmp"
    ## Build directories
    DIR[BUILD,HTML]: "_build"
    DIR[BUILD,LATEX]: "_build_latex"

    # Static .rst files
    DIR[RST]: "../../../docs/report"

    # Working directories
    ## Input data files (.zip, .xml)
    DIR[WORKING,DATA]: "{DIR[WORKING]}/data"
    ## Static source files from git
    DIR[WORKING,SRC]: "{DIR[WORKING]}/src"
    DIR[WORKING,SRC,STATIC]: "{DIR[WORKING,SRC]}/_static"

    # Static html content
    DIR[STATIC]: "{DIR[BUILD,HTML]}/_static"
    DIR[STATIC,VPP]: "{DIR[STATIC]}/vpp"
    DIR[STATIC,DPDK]: "{DIR[STATIC]}/dpdk"
    DIR[STATIC,ARCH]: "{DIR[STATIC]}/archive"

    # Detailed test results
    DIR[DTR]: "{DIR[WORKING,SRC]}/detailed_test_results"
    DIR[DTR,PERF,DPDK]: "{DIR[DTR]}/dpdk_performance_results"
    DIR[DTR,PERF,VPP]: "{DIR[DTR]}/vpp_performance_results"
    DIR[DTR,PERF,HC]: "{DIR[DTR]}/honeycomb_performance_results"
    DIR[DTR,FUNC,VPP]: "{DIR[DTR]}/vpp_functional_results"
    DIR[DTR,FUNC,HC]: "{DIR[DTR]}/honeycomb_functional_results"
    DIR[DTR,FUNC,NSHSFC]: "{DIR[DTR]}/nshsfc_functional_results"
    DIR[DTR,PERF,VPP,IMPRV]: "{DIR[WORKING,SRC]}/vpp_performance_tests/performance_improvements"

    # Detailed test configurations
    DIR[DTC]: "{DIR[WORKING,SRC]}/test_configuration"
    DIR[DTC,PERF,VPP]: "{DIR[DTC]}/vpp_performance_configuration"
    DIR[DTC,FUNC,VPP]: "{DIR[DTC]}/vpp_functional_configuration"

    # Detailed tests operational data
    DIR[DTO]: "{DIR[WORKING,SRC]}/test_operational_data"
    DIR[DTO,PERF,VPP]: "{DIR[DTO]}/vpp_performance_operational_data"

    # .css patch file to fix tables generated by Sphinx
    DIR[CSS_PATCH_FILE]: "{DIR[STATIC]}/theme_overrides.css"
```

(continues on next page)

(continued from previous page)

```
    DIR[CSS_PATCH_FILE2]: "{DIR[WORKING,SRC,STATIC]}/theme_overrides.css"

  urls:
    URL[JENKINS,CSIT]: "https://jenkins.fd.io/view/csit/job"
    URL[JENKINS,HC]: "https://jenkins.fd.io/view/hc2vpp/job"

  make-dirs:
  # List the directories which are created while preparing the environment.
  # All directories MUST be defined in "paths" section.
  - "DIR[WORKING,DATA]"
  - "DIR[STATIC,VPP]"
  - "DIR[STATIC,DPDK]"
  - "DIR[STATIC,ARCH]"
  - "DIR[BUILD,LATEX]"
  - "DIR[WORKING,SRC]"
  - "DIR[WORKING,SRC,STATIC]"

  remove-dirs:
  # List the directories which are deleted while cleaning the environment.
  # All directories MUST be defined in "paths" section.
  #- "DIR[BUILD,HTML]"

  build-dirs:
  # List the directories where the results (build) is stored.
  # All directories MUST be defined in "paths" section.
  - "DIR[BUILD,HTML]"
  - "DIR[BUILD,LATEX]"
```

It is possible to use defined items in the definition of other items, e.g.:

```
DIR[WORKING,DATA]: "{DIR[WORKING]}/data"
```

will be automatically changed to

```
DIR[WORKING,DATA]: "_tmp/data"
```

Section: Configuration

This section specifies the groups of parameters which are repeatedly used in the elements defined later in the specification file. It has the following parts:

- data sets - Specification of data sets used later in element's specifications to define the input data.
- plot layouts - Specification of plot layouts used later in plots' specifications to define the plot layout.

The structure of the section "Configuration" is as follows (example):

```
-
  type: "configuration"
  data-sets:
    plot-vpp-throughput-latency:
      csit-vpp-perf-1710-all:
      - 11
      - 12
      - 13
      - 14
      - 15
      - 16
      - 17
      - 18
```

(continues on next page)

(continued from previous page)

```
      - 19
      - 20
    vpp-perf-results:
      csit-vpp-perf-1710-all:
      - 20
      - 23
  plot-layouts:
    plot-throughput:
      xaxis:
        autorange: True
        autotick: False
        fixedrange: False
        gridcolor: "rgb(238, 238, 238)"
        linecolor: "rgb(238, 238, 238)"
        linewidth: 1
        showgrid: True
        showline: True
        showticklabels: True
        tickcolor: "rgb(238, 238, 238)"
        tickmode: "linear"
        title: "Indexed Test Cases"
        zeroline: False
      yaxis:
        gridcolor: "rgb(238, 238, 238)'"
        hoverformat: ".4s"
        linecolor: "rgb(238, 238, 238)"
        linewidth: 1
        range: []
        showgrid: True
        showline: True
        showticklabels: True
        tickcolor: "rgb(238, 238, 238)"
        title: "Packets Per Second [pps]"
        zeroline: False
      boxmode: "group"
      boxgroupgap: 0.5
      autosize: False
      margin:
        t: 50
        b: 20
        l: 50
        r: 20
      showlegend: True
      legend:
        orientation: "h"
      width: 700
      height: 1000
```

The definitions from this sections are used in the elements, e.g.:

```
-
  type: "plot"
  title: "VPP Performance 64B-1t1c-(eth|dot1q|dot1ad)-(l2xcbase|l2bdbasemaclrn)-ndrdisc"
  algorithm: "plot_performance_box"
  output-file-type: ".html"
  output-file: "{DIR[STATIC,VPP]}/64B-1t1c-l2-sel1-ndrdisc"
  data:
    "plot-vpp-throughput-latency"
  filter: "'64B' and ('BASE' or 'SCALE') and 'NDRDISC' and '1T1C' and ('L2BDMACSTAT' or 'L2BDMACLRN
↪' or 'L2XCFWD') and not 'VHOST'"
  parameters:
```

(continues on next page)

(continued from previous page)

```
  - "throughput"
  - "parent"
  traces:
    hoverinfo: "x+y"
    boxpoints: "outliers"
    whiskerwidth: 0
  layout:
    title: "64B-1t1c-(eth|dot1q|dot1ad)-(l2xcbase|l2bdbasemaclrn)-ndrdisc"
    layout:
      "plot-throughput"
```

Section: Debug mode

This section is optional as it configures the debug mode. It is used if one does not want to download input data files and use local files instead.

If the debug mode is configured, the "input" section is ignored.

This section has the following parts:

- type: "debug" - says that this is the section "debug".
- general:
  - input-format - xml or zip.
  - extract - if "zip" is defined as the input format, this file is extracted from the zip file, otherwise this parameter is ignored.
- builds - list of builds from which the data is used. Must include a job name as a key and then a list of builds and their output files.

The structure of the section "Debug" is as follows (example):

```
-
  type: "debug"
  general:
    input-format: "zip"  # zip or xml
    extract: "robot-plugin/output.xml"  # Only for zip
  builds:
    # The files must be in the directory DIR[WORKING,DATA]
    csit-dpdk-perf-1707-all:
    -
      build: 10
      file: "csit-dpdk-perf-1707-all__10.xml"
    -
      build: 9
      file: "csit-dpdk-perf-1707-all__9.xml"
    csit-nsh_sfc-verify-func-1707-ubuntu1604-virl:
    -
      build: 2
      file: "csit-nsh_sfc-verify-func-1707-ubuntu1604-virl-2.xml"
    csit-vpp-functional-1707-ubuntu1604-virl:
    -
      build: lastSuccessfulBuild
      file: "csit-vpp-functional-1707-ubuntu1604-virl-lastSuccessfulBuild.xml"
    hc2vpp-csit-integration-1707-ubuntu1604:
    -
      build: lastSuccessfulBuild
      file: "hc2vpp-csit-integration-1707-ubuntu1604-lastSuccessfulBuild.xml"
    csit-vpp-perf-1707-all:
    -
```

(continues on next page)

(continued from previous page)

```
    build: 16
    file: "csit-vpp-perf-1707-all__16__output.xml"
  -
    build: 17
    file: "csit-vpp-perf-1707-all__17__output.xml"
```

Section: Static

This section defines the static content which is stored in git and will be used as a source to generate the report.

This section has these parts:

- type: "static" - says that this section is the "static".
- src-path - path to the static content.
- dst-path - destination path where the static content is copied and then processed.

::

  - type: "static" src-path: "{DIR[RST]}" dst-path: "{DIR[WORKING,SRC]}"

Section: Input

This section defines the data used to generate elements. It is mandatory if the debug mode is not used.

This section has the following parts:

- type: "input" - says that this section is the "input".
- general - parameters common to all builds:
  - file-name: file to be downloaded.
  - file-format: format of the downloaded file, ".zip" or ".xml" are supported.
  - download-path: path to be added to url pointing to the file, e.g.: "{job}/{build}/robot/report/*zip*/{filename}"; {job}, {build} and {filename} are replaced by proper values defined in this section.
  - extract: file to be extracted from downloaded zip file, e.g.: "output.xml"; if xml file is downloaded, this parameter is ignored.
- builds - list of jobs (keys) and numbers of builds which output data will be downloaded.

The structure of the section "Input" is as follows (example from 17.07 report):

```
-
  type: "input"  # Ignored in debug mode
  general:
    file-name: "robot-plugin.zip"
    file-format: ".zip"
    download-path: "{job}/{build}/robot/report/*zip*/{filename}"
    extract: "robot-plugin/output.xml"
  builds:
    csit-vpp-perf-1707-all:
    - 9
    - 10
    - 13
    - 14
    - 15
    - 16
```

(continues on next page)

(continued from previous page)

```
    - 17
    - 18
    - 19
    - 21
    - 22
    csit-dpdk-perf-1707-all:
    - 1
    - 2
    - 3
    - 4
    - 5
    - 6
    - 7
    - 8
    - 9
    - 10
    csit-vpp-functional-1707-ubuntu1604-virl:
    - lastSuccessfulBuild
    hc2vpp-csit-perf-master-ubuntu1604:
    - 8
    - 9
    hc2vpp-csit-integration-1707-ubuntu1604:
    - lastSuccessfulBuild
    csit-nsh_sfc-verify-func-1707-ubuntu1604-virl:
    - 2
```

Section: Output

This section specifies which format(s) will be generated (html, pdf) and which versions will be generated for each format.

This section has the following parts:

- type: "output" - says that this section is the "output".
- format: html or pdf.
- version: defined for each format separately.

The structure of the section "Output" is as follows (example):

```
-
  type: "output"
  format:
    html:
    - full
    pdf:
    - full
    - minimal
```

TODO: define the names of versions

Content of "minimal" version

TODO: define the name and content of this version

Section: Table

This section defines a table to be generated. There can be 0 or more "table" sections.

This section has the following parts:

- type: "table" - says that this section defines a table.
- title: Title of the table.
- algorithm: Algorithm which is used to generate the table. The other parameters in this section must provide all information needed by the used algorithm.
- template: (optional) a .csv file used as a template while generating the table.
- output-file-ext: extension of the output file.
- output-file: file which the table will be written to.
- columns: specification of table columns:
  - title: The title used in the table header.
  - data: Specification of the data, it has two parts - command and arguments:
    - command:
      - template - take the data from template, arguments:
      - number of column in the template.
      - data - take the data from the input data, arguments:
      - jobs and builds which data will be used.
      - operation - performs an operation with the data already in the table, arguments:
      - operation to be done, e.g.: mean, stdev, relative_change (compute the relative change between two columns) and display number of data samples ~= number of test jobs. The operations are implemented in the utils.py TODO: Move from utils,py to e.g. operations.py
      - numbers of columns which data will be used (optional).
- data: Specify the jobs and builds which data is used to generate the table.
- filter: filter based on tags applied on the input data, if "template" is used, filtering is based on the template.
- parameters: Only these parameters will be put to the output data structure.

The structure of the section "Table" is as follows (example of "table_performance_improvements"):

```
-
  type: "table"
  title: "Performance improvements"
  algorithm: "table_performance_improvements"
  template: "{DIR[DTR,PERF,VPP,IMPRV]}/tmpl_performance_improvements.csv"
  output-file-ext: ".csv"
  output-file: "{DIR[DTR,PERF,VPP,IMPRV]}/performance_improvements"
  columns:
  -
    title: "VPP Functionality"
    data: "template 1"
  -
    title: "Test Name"
    data: "template 2"
  -
    title: "VPP-16.09 mean [Mpps]"
    data: "template 3"
  -
    title: "VPP-17.01 mean [Mpps]"
    data: "template 4"
  -
```

(continues on next page)

(continued from previous page)

```
    title: "VPP-17.04 mean [Mpps]"
    data: "template 5"
  -
    title: "VPP-17.07 mean [Mpps]"
    data: "data csit-vpp-perf-1707-all mean"
  -
    title: "VPP-17.07 stdev [Mpps]"
    data: "data csit-vpp-perf-1707-all stdev"
  -
    title: "17.04 to 17.07 change [%]"
    data: "operation relative_change 5 4"
  data:
    csit-vpp-perf-1707-all:
    - 9
    - 10
    - 13
    - 14
    - 15
    - 16
    - 17
    - 18
    - 19
    - 21
  filter: "template"
  parameters:
  - "throughput"
```

Example of "table_details" which generates "Detailed Test Results - VPP Performance Results":

```
-
  type: "table"
  title: "Detailed Test Results - VPP Performance Results"
  algorithm: "table_details"
  output-file-ext: ".csv"
  output-file: "{DIR[WORKING]}/vpp_performance_results"
  columns:
  -
    title: "Name"
    data: "data test_name"
  -
    title: "Documentation"
    data: "data test_documentation"
  -
    title: "Status"
    data: "data test_msg"
  data:
    csit-vpp-perf-1707-all:
    - 17
  filter: "all"
  parameters:
  - "parent"
  - "doc"
  - "msg"
```

Example of "table_details" which generates "Test configuration - VPP Performance Test Configs":

```
-
  type: "table"
  title: "Test configuration - VPP Performance Test Configs"
  algorithm: "table_details"
  output-file-ext: ".csv"
```

(continues on next page)

(continued from previous page)

```
  output-file: "{DIR[WORKING]}/vpp_test_configuration"
  columns:
  -
    title: "Name"
    data: "data name"
  -
    title: "VPP API Test (VAT) Commands History - Commands Used Per Test Case"
    data: "data show-run"
  data:
    csit-vpp-perf-1707-all:
    - 17
  filter: "all"
  parameters:
  - "parent"
  - "name"
  - "show-run"
```

Section: Plot

This section defines a plot to be generated. There can be 0 or more "plot" sections.

This section has these parts:

- type: "plot" - says that this section defines a plot.
- title: Plot title used in the logs. Title which is displayed is in the section "layout".
- output-file-type: format of the output file.
- output-file: file which the plot will be written to.
- algorithm: Algorithm used to generate the plot. The other parameters in this section must provide all information needed by plot.ly to generate the plot. For example:
  - traces
  - layout
  - These parameters are transparently passed to plot.ly.
- data: Specify the jobs and numbers of builds which data is used to generate the plot.
- filter: filter applied on the input data.
- parameters: Only these parameters will be put to the output data structure.

The structure of the section "Plot" is as follows (example of a plot showing throughput in a chart box-with-whiskers):

```
-
  type: "plot"
  title: "VPP Performance 64B-1t1c-(eth|dot1q|dot1ad)-(l2xcbase|l2bdbasemaclrn)-ndrdisc"
  algorithm: "plot_performance_box"
  output-file-type: ".html"
  output-file: "{DIR[STATIC,VPP]}/64B-1t1c-l2-sel1-ndrdisc"
  data:
    csit-vpp-perf-1707-all:
    - 9
    - 10
    - 13
    - 14
    - 15
    - 16
    - 17
```

(continues on next page)

(continued from previous page)

```
    - 18
    - 19
    - 21
  # Keep this formatting, the filter is enclosed with " (quotation mark) and
  # each tag is enclosed with ' (apostrophe).
  filter: "'64B' and 'BASE' and 'NDRDISC' and '1T1C' and ('L2BDMACSTAT' or 'L2BDMACLRN' or 'L2XCFWD
↪') and not 'VHOST'"
  parameters:
  - "throughput"
  - "parent"
  traces:
    hoverinfo: "x+y"
    boxpoints: "outliers"
    whiskerwidth: 0
  layout:
    title: "64B-1t1c-(eth|dot1q|dot1ad)-(l2xcbase|l2bdbasemaclrn)-ndrdisc"
    xaxis:
      autorange: True
      autotick: False
      fixedrange: False
      gridcolor: "rgb(238, 238, 238)"
      linecolor: "rgb(238, 238, 238)"
      linewidth: 1
      showgrid: True
      showline: True
      showticklabels: True
      tickcolor: "rgb(238, 238, 238)"
      tickmode: "linear"
      title: "Indexed Test Cases"
      zeroline: False
    yaxis:
      gridcolor: "rgb(238, 238, 238)'"
      hoverformat: ".4s"
      linecolor: "rgb(238, 238, 238)"
      linewidth: 1
      range: []
      showgrid: True
      showline: True
      showticklabels: True
      tickcolor: "rgb(238, 238, 238)"
      title: "Packets Per Second [pps]"
      zeroline: False
    boxmode: "group"
    boxgroupgap: 0.5
    autosize: False
    margin:
      t: 50
      b: 20
      l: 50
      r: 20
    showlegend: True
    legend:
      orientation: "h"
    width: 700
    height: 1000
```

The structure of the section "Plot" is as follows (example of a plot showing latency in a box chart):

```
-
  type: "plot"
  title: "VPP Latency 64B-1t1c-(eth|dot1q|dot1ad)-(l2xcbase|l2bdbasemaclrn)-ndrdisc"
```

(continues on next page)

(continued from previous page)

```
  algorithm: "plot_latency_box"
  output-file-type: ".html"
  output-file: "{DIR[STATIC,VPP]}/64B-1t1c-l2-sel1-ndrdisc-lat50"
  data:
    csit-vpp-perf-1707-all:
    - 9
    - 10
    - 13
    - 14
    - 15
    - 16
    - 17
    - 18
    - 19
    - 21
  filter: "'64B' and 'BASE' and 'NDRDISC' and '1T1C' and ('L2BDMACSTAT' or 'L2BDMACLRN' or 'L2XCFWD
↪') and not 'VHOST'"
  parameters:
  - "latency"
  - "parent"
  traces:
    boxmean: False
  layout:
    title: "64B-1t1c-(eth|dot1q|dot1ad)-(l2xcbase|l2bdbasemaclrn)-ndrdisc"
    xaxis:
      autorange: True
      autotick: False
      fixedrange: False
      gridcolor: "rgb(238, 238, 238)"
      linecolor: "rgb(238, 238, 238)"
      linewidth: 1
      showgrid: True
      showline: True
      showticklabels: True
      tickcolor: "rgb(238, 238, 238)"
      tickmode: "linear"
      title: "Indexed Test Cases"
      zeroline: False
    yaxis:
      gridcolor: "rgb(238, 238, 238)'"
      hoverformat: ""
      linecolor: "rgb(238, 238, 238)"
      linewidth: 1
      range: []
      showgrid: True
      showline: True
      showticklabels: True
      tickcolor: "rgb(238, 238, 238)"
      title: "Latency min/avg/max [uSec]"
      zeroline: False
    boxmode: "group"
    boxgroupgap: 0.5
    autosize: False
    margin:
      t: 50
      b: 20
      l: 50
      r: 20
    showlegend: True
    legend:
      orientation: "h"
```

(continues on next page)

(continued from previous page)

```
    width: 700
    height: 1000
```

The structure of the section "Plot" is as follows (example of a plot showing VPP HTTP server performance in a box chart with pre-defined data "plot-vpp-httlp-server-performance" set and plot layout "plot-cps"):

```
-
  type: "plot"
  title: "VPP HTTP Server Performance"
  algorithm: "plot_http_server_performance_box"
  output-file-type: ".html"
  output-file: "{DIR[STATIC,VPP]}/http-server-performance-cps"
  data:
    "plot-vpp-httlp-server-performance"
  # Keep this formatting, the filter is enclosed with " (quotation mark) and
  # each tag is enclosed with ' (apostrophe).
  filter: "'HTTP' and 'TCP_CPS'"
  parameters:
  - "result"
  - "name"
  traces:
    hoverinfo: "x+y"
    boxpoints: "outliers"
    whiskerwidth: 0
  layout:
    title: "VPP HTTP Server Performance"
    layout:
      "plot-cps"
```

Section: file

This section defines a file to be generated. There can be 0 or more "file" sections.

This section has the following parts:

- type: "file" - says that this section defines a file.
- title: Title of the table.
- algorithm: Algorithm which is used to generate the file. The other parameters in this section must provide all information needed by the used algorithm.
- output-file-ext: extension of the output file.
- output-file: file which the file will be written to.
- file-header: The header of the generated .rst file.
- dir-tables: The directory with the tables.
- data: Specify the jobs and builds which data is used to generate the table.
- filter: filter based on tags applied on the input data, if "all" is used, no filtering is done.
- parameters: Only these parameters will be put to the output data structure.
- chapters: the hierarchy of chapters in the generated file.
- start-level: the level of the the top-level chapter.

The structure of the section "file" is as follows (example):

```
-
  type: "file"
  title: "VPP Performance Results"
  algorithm: "file_test_results"
  output-file-ext: ".rst"
  output-file: "{DIR[DTR,PERF,VPP]}/vpp_performance_results"
  file-header: "\n.. |br| raw:: html\n\n    <br />\n\n\n.. |prein| raw:: html\n\n    <pre>\n\n\n..␣
↪|preout| raw:: html\n\n    </pre>\n\n"
  dir-tables: "{DIR[DTR,PERF,VPP]}"
  data:
    csit-vpp-perf-1707-all:
    - 22
  filter: "all"
  parameters:
  - "name"
  - "doc"
  - "level"
  data-start-level: 2  # 0, 1, 2, ...
  chapters-start-level: 2  # 0, 1, 2, ...
```

Static content

- Manually created / edited files.
- .rst files, static .csv files, static pictures (.svg), …
- Stored in CSIT git repository.

No more details about the static content in this document.

Data to process

The PAL processes tests results and other information produced by Jenkins jobs. The data are now stored as robot results in Jenkins (TODO: store the data in nexus) either as .zip and / or .xml files.

6.3.3 Data processing

As the first step, the data are downloaded and stored locally (typically on a Jenkins slave). If .zip files are used, the given .xml files are extracted for further processing.

Parsing of the .xml files is performed by a class derived from "robot.api.ResultVisitor", only necessary methods are overridden. All and only necessary data is extracted from .xml file and stored in a structured form.

The parsed data are stored as the multi-indexed pandas.Series data type. Its structure is as follows:

```
<job name>
  <build>
    <metadata>
    <suites>
    <tests>
```

"job name", "build", "metadata", "suites", "tests" are indexes to access the data. For example:

```
data =

job 1 name:
  build 1:
    metadata: metadata
```

(continues on next page)

(continued from previous page)

```
    suites: suites
    tests: tests
  ...
  build N:
    metadata: metadata
    suites: suites
    tests: tests
...
job M name:
  build 1:
    metadata: metadata
    suites: suites
    tests: tests
  ...
  build N:
    metadata: metadata
    suites: suites
    tests: tests
```

Using indexes data["job 1 name"]["build 1"]["tests"] (e.g.: data["csit-vpp-perf-1704-all"]["17"]["tests"]) we get a list of all tests with all tests data.

Data will not be accessible directly using indexes, but using getters and filters.

Structure of metadata:

```
"metadata": {
    "version": "VPP version",
    "job": "Jenkins job name"
    "build": "Information about the build"
},
```

Structure of suites:

```
"suites": {
    "Suite name 1": {
        "doc": "Suite 1 documentation"
        "parent": "Suite 1 parent"
    }
    "Suite name N": {
        "doc": "Suite N documentation"
        "parent": "Suite N parent"
    }
```

Structure of tests:

Performance tests:

```
"tests": {
    "ID": {
        "name": "Test name",
        "parent": "Name of the parent of the test",
        "doc": "Test documentation"
        "msg": "Test message"
        "tags": ["tag 1", "tag 2", "tag n"],
        "type": "PDR" | "NDR",
        "throughput": {
            "value": int,
            "unit": "pps" | "bps" | "percentage"
        },
        "latency": {
            "direction1": {
```

(continues on next page)

(continued from previous page)

```
                "100": {
                    "min": int,
                    "avg": int,
                    "max": int
                },
                "50": {  # Only for NDR
                    "min": int,
                    "avg": int,
                    "max": int
                },
                "10": {  # Only for NDR
                    "min": int,
                    "avg": int,
                    "max": int
                }
            },
            "direction2": {
                "100": {
                    "min": int,
                    "avg": int,
                    "max": int
                },
                "50": {  # Only for NDR
                    "min": int,
                    "avg": int,
                    "max": int
                },
                "10": {  # Only for NDR
                    "min": int,
                    "avg": int,
                    "max": int
                }
            }
        },
        "lossTolerance": "lossTolerance"  # Only for PDR
        "vat-history": "DUT1 and DUT2 VAT History"
        },
        "show-run": "Show Run"
    },
    "ID" {
        # next test
    }
```

Functional tests:

```
"tests": {
    "ID": {
        "name": "Test name",
        "parent": "Name of the parent of the test",
        "doc": "Test documentation"
        "msg": "Test message"
        "tags": ["tag 1", "tag 2", "tag n"],
        "vat-history": "DUT1 and DUT2 VAT History"
        "show-run": "Show Run"
        "status": "PASS" | "FAIL"
    },
    "ID" {
        # next test
    }
}
```

Note: ID is the lowercase full path to the test.

Data filtering

The first step when generating an element is getting the data needed to construct the element. The data are filtered from the processed input data.

The data filtering is based on:

- job name(s).
- build number(s).
- tag(s).
- required data - only this data is included in the output.

WARNING: The filtering is based on tags, so be careful with tagging.

For example, the element which specification includes:

```
data:
  csit-vpp-perf-1707-all:
  - 9
  - 10
  - 13
  - 14
  - 15
  - 16
  - 17
  - 18
  - 19
  - 21
filter:
  - "'64B' and 'BASE' and 'NDRDISC' and '1T1C' and ('L2BDMACSTAT' or 'L2BDMACLRN' or 'L2XCFWD') and␣
↪not 'VHOST'"
```

will be constructed using data from the job "csit-vpp-perf-1707-all", for all listed builds and the tests with the list of tags matching the filter conditions.

The output data structure for filtered test data is:

```
- job 1
  - build 1
    - test 1
      - parameter 1
      - parameter 2
      ...
      - parameter n
    ...
    - test n
    ...
  ...
  - build n
...
- job n
```

Data analytics

Data analytics part implements:

- methods to compute statistical data from the filtered input data.
- trending.

Throughput Speedup Analysis - Multi-Core with Multi-Threading

Throughput Speedup Analysis (TSA) calculates throughput speedup ratios for tested 1-, 2- and 4-core multi-threaded VPP configurations using the following formula:

```
                            N_core_throughput
N_core_throughput_speedup = ------------------
                            1_core_throughput
```

Multi-core throughput speedup ratios are plotted in grouped bar graphs for throughput tests with 64B/78B frame size, with number of cores on X-axis and speedup ratio on Y-axis.

For better comparison multiple test results' data sets are plotted per each graph:

- graph type: grouped bars;
- graph X-axis: (testcase index, number of cores);
- graph Y-axis: speedup factor.

Subset of existing performance tests is covered by TSA graphs.

Model for TSA:

```
-
  type: "plot"
  title: "TSA: 64B-*-(eth|dot1q|dot1ad)-(l2xcbase|l2bdbasemaclrn)-ndrdisc"
  algorithm: "plot_throughput_speedup_analysis"
  output-file-type: ".html"
  output-file: "{DIR[STATIC,VPP]}/10ge2p1x520-64B-l2-tsa-ndrdisc"
  data:
    "plot-throughput-speedup-analysis"
  filter: "'NIC_Intel-X520-DA2' and '64B' and 'BASE' and 'NDRDISC' and ('L2BDMACSTAT' or 'L2BDMACLRN
↪' or 'L2XCFWD') and not 'VHOST'"
  parameters:
  - "throughput"
  - "parent"
  - "tags"
  layout:
    title: "64B-*-(eth|dot1q|dot1ad)-(l2xcbase|l2bdbasemaclrn)-ndrdisc"
    layout:
      "plot-throughput-speedup-analysis"
```

Comparison of results from two sets of the same test executions

This algorithm enables comparison of results coming from two sets of the same test executions. It is used to quantify performance changes across all tests after test environment changes e.g. Operating System upgrades/patches, Hardware changes.

It is assumed that each set of test executions includes multiple runs of the same tests, 10 or more, to verify test results repeatibility and to yield statistically meaningful results data.

Comparison results are presented in a table with a specified number of the best and the worst relative changes between the two sets. Following table columns are defined:

- name of the test;
- throughput mean values of the reference set;
- throughput standard deviation of the reference set;
- throughput mean values of the set to compare;
- throughput standard deviation of the set to compare;

- relative change of the mean values.

The model

The model specifies:

- type: "table" - means this section defines a table.
- title: Title of the table.
- algorithm: Algorithm which is used to generate the table. The other parameters in this section must provide all information needed by the used algorithm.
- output-file-ext: Extension of the output file.
- output-file: File which the table will be written to.
- reference - the builds which are used as the reference for comparison.
- compare - the builds which are compared to the reference.
- data: Specify the sources, jobs and builds, providing data for generating the table.
- filter: Filter based on tags applied on the input data, if "template" is used, filtering is based on the template.
- parameters: Only these parameters will be put to the output data structure.
- nr-of-tests-shown: Number of the best and the worst tests presented in the table. Use 0 (zero) to present all tests.

Example:

```
-
  type: "table"
  title: "Performance comparison"
  algorithm: "table_performance_comparison"
  output-file-ext: ".csv"
  output-file: "{DIR[DTR,PERF,VPP,IMPRV]}/vpp_performance_comparison"
  reference:
    title: "csit-vpp-perf-1801-all - 1"
    data:
      csit-vpp-perf-1801-all:
      - 1
      - 2
  compare:
    title: "csit-vpp-perf-1801-all - 2"
    data:
      csit-vpp-perf-1801-all:
      - 1
      - 2
  data:
    "vpp-perf-comparison"
  filter: "all"
  parameters:
  - "name"
  - "parent"
  - "throughput"
  nr-of-tests-shown: 20
```

Advanced data analytics

In the future advanced data analytics (ADA) will be added to analyze the telemetry data collected from SUT telemetry sources and correlate it to performance test results.

TODO

- describe the concept of ADA.
- add specification.

6.3.4 Data presentation

Generates the plots and tables according to the report models per specification file. The elements are generated using algorithms and data specified in their models.

Tables

- tables are generated by algorithms implemented in PAL, the model includes the algorithm and all necessary information.
- output format: csv
- generated tables are stored in specified directories and linked to .rst files.

Plots

- plot.ly[113] is currently used to generate plots, the model includes the type of plot and all the necessary information to render it.
- output format: html.
- generated plots are stored in specified directories and linked to .rst files.

6.3.5 Report generation

Report is generated using Sphinx and Read_the_Docs template. PAL generates html and pdf formats. It is possible to define the content of the report by specifying the version (TODO: define the names and content of versions).

The process

1. Read the specification.
2. Read the input data.
3. Process the input data.
4. For element (plot, table, file) defined in specification:
   (a) Get the data needed to construct the element using a filter.
   (b) Generate the element.
   (c) Store the element.
5. Generate the report.
6. Store the report (Nexus).

The process is model driven. The elements' models (tables, plots, files and report itself) are defined in the specification file. Script reads the elements' models from specification file and generates the elements.

It is easy to add elements to be generated in the report. If a new type of an element is required, only a new algorithm needs to be implemented and integrated.

[113] https://plot.ly/

6.3.6 Continuous Performance Measurements and Trending

Performance analysis and trending execution sequence:

CSIT PA runs performance analysis, change detection and trending using specified trend analysis metrics over the rolling window of last <N> sets of historical measurement data. PA is defined as follows:

1. PA job triggers:
   (a) By PT job at its completion.
   (b) Manually from Jenkins UI.
2. Download and parse archived historical data and the new data:
   (a) New data from latest PT job is evaluated against the rolling window of <N> sets of historical data.
   (b) Download RF output.xml files and compressed archived data.
   (c) Parse out the data filtering test cases listed in PA specification (part of CSIT PAL specification file).
3. Calculate trend metrics for the rolling window of <N> sets of historical data:
   (a) Calculate quartiles Q1, Q2, Q3.
   (b) Trim outliers using IQR.
   (c) Calculate TMA and TMSD.
   (d) Calculate normal trending range per test case based on TMA and TMSD.
4. Evaluate new test data against trend metrics:
   (a) If within the range of (TMA +/- 3*TMSD) => Result = Pass, Reason = Normal.
   (b) If below the range => Result = Fail, Reason = Regression.
   (c) If above the range => Result = Pass, Reason = Progression.
5. Generate and publish results
   (a) Relay evaluation result to job result.
   (b) Generate a new set of trend analysis summary graphs and drill-down graphs.
      i. Summary graphs to include measured values with Normal, Progression and Regression markers. MM shown in the background if possible.
      ii. Drill-down graphs to include MM, TMA and TMSD.
   (c) Publish trend analysis graphs in html format on https://docs.fd.io/csit/master/trending/.

Parameters to specify:

- job to be monitored - the Jenkins job which results are used as input data for this test;
- builds used for trending plot(s) - specified by a list of build numbers or by a range of builds defined by the first and the last buld number;
- list plots to generate:
  - plot title;
  - output file name;
  - data for plots;
  - tests to be displayed in the plot defined by a filter;
  - list of parameters to extract from the data;

- periods (daily = 1, weekly = 5, monthly = 30);
- plot layout

Example:

```
-
  type: "cpta"
  title: "Continuous Performance Trending and Analysis"
  algorithm: "cpta"
  output-file-type: ".html"
  output-file: "{DIR[STATIC,VPP]}/cpta"
  data: "plot-performance-trending"
  plots:
    - title: "VPP 1T1C L2 64B Packet Throughput - {period} Trending"
      output-file-name: "l2-1t1c-x520"
      data: "plot-performance-trending"
      filter: "'NIC_Intel-X520-DA2' and 'MRR' and '64B' and ('BASE' or 'SCALE') and '1T1C' and (
↪'L2BDMACSTAT' or 'L2BDMACLRN' or 'L2XCFWD') and not 'VHOST' and not 'MEMIF'"
      parameters:
      - "result"
#      - "name"
      periods:
      - 1
      - 5
      - 30
      layout: "plot-cpta"

    - title: "VPP 2T2C L2 64B Packet Throughput - {period} Trending"
      output-file-name: "l2-2t2c-x520"
      data: "plot-performance-trending"
      filter: "'NIC_Intel-X520-DA2' and 'MRR' and '64B' and ('BASE' or 'SCALE') and '2T2C' and (
↪'L2BDMACSTAT' or 'L2BDMACLRN' or 'L2XCFWD') and not 'VHOST' and not 'MEMIF'"
      parameters:
      - "result"
#      - "name"
      periods:
      - 1
      - 5
      - 30
      layout: "plot-cpta"
```

6.3.7 API

List of modules, classes, methods and functions

```
specification_parser.py

    class Specification

        Methods:
            read_specification
            set_input_state
            set_input_file_name

        Getters:
            specification
            environment
            debug
            is_debug
            input
```

(continues on next page)

(continued from previous page)

```
            builds
            output
            tables
            plots
            files
            static


input_data_parser.py

    class InputData

        Methods:
            read_data
            filter_data

        Getters:
            data
            metadata
            suites
            tests


environment.py

    Functions:
        clean_environment

    class Environment

        Methods:
            set_environment

        Getters:
            environment


input_data_files.py

    Functions:
        download_data_files
        unzip_files


generator_tables.py

    Functions:
        generate_tables

    Functions implementing algorithms to generate particular types of
    tables (called by the function "generate_tables"):
        table_details
        table_performance_improvements


generator_plots.py

    Functions:
        generate_plots

    Functions implementing algorithms to generate particular types of
```

(continues on next page)

(continued from previous page)

```
    plots (called by the function "generate_plots"):
        plot_performance_box
        plot_latency_box


generator_files.py

    Functions:
        generate_files

    Functions implementing algorithms to generate particular types of
    files (called by the function "generate_files"):
        file_test_results


report.py

    Functions:
        generate_report

    Functions implementing algorithms to generate particular types of
    report (called by the function "generate_report"):
        generate_html_report
        generate_pdf_report

    Other functions called by the function "generate_report":
        archive_input_data
        archive_report
```

PAL functional diagram

pal_func_diagram.pdf

How to add an element

Element can be added by adding it's model to the specification file. If the element is to be generated by an existing algorithm, only it's parameters must be set.

If a brand new type of element needs to be added, also the algorithm must be implemented. Element generation algorithms are implemented in the files with names starting with "generator" prefix. The name of the function implementing the algorithm and the name of algorithm in the specification file have to be the same.

6.4 CSIT TAGs Descriptions

All CSIT test cases are labelled with Robot Framework tags used to allow for easy test case type identification, test case grouping and selection for execution. Following sections list currently used CSIT TAGs and their documentation based on the content of tag documentation rst file[114].

6.4.1 Topology TAGs

3_NODE_DOUBLE_LINK_TOPO

3 nodes connected in a circular topology with two links interconnecting the devices.

3_NODE_SINGLE_LINK_TOPO

3 nodes connected in a circular topology with at least one link interconnecting devices.

6.4.2 Objective TAGs

SKIP_PATCH

Test case(s) marked to not run in case of vpp-csit-verify (i.e. VPP patch) and csit-vpp-verify jobs (i.e. CSIT patch).

SKIP_VPP_PATCH

Test case(s) marked to not run in case of vpp-csit-verify (i.e. VPP patch).

6.4.3 Environment TAGs

HW_ENV

DUTs and TGs are running on bare metal.

VM_ENV

DUTs and TGs are running in virtual environment.

VPP_VM_ENV

DUTs with VPP and capable of running Virtual Machine.

6.4.4 NIC model tags

NIC_Intel-X520-DA2

Intel X520-DA2 NIC.

[114] https://git.fd.io/csit/tree/docs/tag_documentation.rst?h=rls1801_2

NIC_Intel-XL710

Intel XL710 NIC.

NIC_Intel-X710

Intel X710 NIC.

NIC_Cisco-VIC-1227

VIC-1227 by Cisco.

NIC_Cisco-VIC-1385

VIC-1385 by Cisco.

6.4.5 Scaling TAGs

FIB_20K

2x10,000 entries in single fib table

FIB_200K

2x100,000 entries in single fib table

FIB_2M

2x1,000,000 entries in single fib table

TNL_1000

IPSec in tunnel mode - 1000 tunnels.

SRC_USER_10

Traffic flow with 10 unique IPs (users) in one direction.

SRC_USER_100

Traffic flow with 100 unique IPs (users) in one direction.

SRC_USER_1000

Traffic flow with 1000 unique IPs (users) in one direction.

SRC_USER_2000

Traffic flow with 2000 unique IPs (users) in one direction.

SRC_USER_4000

Traffic flow with 4000 unique IPs (users) in one direction.

100_FLOWS

Traffic stream with 100 unique flows (10 IPs/users x 10 UDP ports) in one direction.

10k_FLOWS

Traffic stream with 10 000 unique flows (10 IPs/users x 1000 UDP ports) in one direction.

100k_FLOWS

Traffic stream with 100 000 unique flows (100 IPs/users x 1000 UDP ports) in one direction.

6.4.6 Tags marking functional vs. performance of tests

FUNCTEST

All functional test cases.

PERFTEST

All performance test cases.

6.4.7 Performance testing tags

PDRDISC

Partial Drop Rate evaluation of single run result, with non-zero packet loss tolerance (LT) expressed in percentage of packets transmitted.

NDRDISC

Non Drop Rate evaluation of results. Loss acceptance of dropped packets is set to zero lost packets.

NDRCHK

Performance tests where TG verifies DUTs' throughput at ref-NDR (reference Non Drop Rate) with zero packet loss tolerance.

PDRCHK

Performance tests where TG verifies DUTs' throughput at ref-PDR (reference Partial Drop Rate) with 0.5% loss tolerance.

MRR

Performance tests where TG sends the traffic at maximum rate (line rate) and reports total sent/received packets over performance trial duration.

NDRPDRDISC

Find performance of DUT based on RFC 2544[115] with linear / binary / combined search. (Previous LONG tests.)

[115] https://tools.ietf.org/html/rfc2544.html

6.4.8 Ethernet frame size tags for performance tests

64B

64B frames used for test.

78B

78B frames used for test.

IMIX

IMIX frame sequence (28x 64B, 16x 570B, 4x 1518B) used for test.

1460B

1460B frames used for test.

1480B

1480B frames used for test.

1514B

1514B frames used for test.

1518B

1518B frames used for test.

9000B

9000B frames used for test.

6.4.9 Test type tags

BASE

Baseline test cases, no encapsulation, no feature(s) configured in tests.

IP4BASE

IPv4 baseline test cases, no encapsulation, no feature(s) configured in tests.

IP6BASE

IPv6 baseline test cases, no encapsulation, no feature(s) configured in tests.

L2XCBASE

L2XC baseline test cases, no encapsulation, no feature(s) configured in tests.

L2BDBASE

L2BD baseline test cases, no encapsulation, no feature(s) configured in tests.

SCALE

Scale test cases.

ENCAP

Test cases where encapsulation is used. Use also encapsulation tag(s).

FEATURE

At least one feature is configured in test cases. Use also feature tag(s).

TLDK

Functional test cases for TLDK.

TCP

Tests which use TCP.

TCP_CPS

Performance tests which measure connections per second using http requests.

TCP_RPS

Performance tests which measure requests per second using http requests.

HTTP

Tests which use HTTP.

6.4.10 Forwarding mode tags

L2BDMACSTAT

VPP L2 bridge-domain, L2 MAC static.

L2BDMACLRN

VPP L2 bridge-domain, L2 MAC learning.

L2XCFWD

VPP L2 point-to-point cross-connect.

IP4FWD

VPP IPv4 routed forwarding.

IP6FWD

VPP IPv6 routed forwarding.

6.4.11 Underlay tags

IP4UNRLAY

IPv4 underlay.

IP6UNRLAY

IPv6 underlay.

MPLSUNRLAY

MPLS underlay.

6.4.12 Overlay tags

L2OVRLAY

L2 overlay.

IP4OVRLAY

IPv4 overlay (IPv4 payload).

IP6OVRLAY

IPv6 overlay (IPv6 payload).

6.4.13 Tagging tags

DOT1Q

All test cases with dot1q.

DOT1AD

All test cases with dot1ad.

6.4.14 Encapsulation tags

ETH

All test cases with base Ethernet (no encapsulation).

LISP

All test cases with LISP.

LISPGPE

All test cases with LISP-GPE.

VXLAN

All test cases with Vxlan.

VXLANGPE

All test cases with VXLAN-GPE.

GRE

All test cases with GRE.

IPSEC

All test cases with IPSEC.

6.4.15 Interface tags

PHY

All test cases which use physical interface(s).

VHOST

All test cases which uses VHOST.

VHOST_256

All test cases which uses VHOST with qemu queue size set to 256.

VHOST_1024

All test cases which uses VHOST with qemu queue size set to 1024.

CFS_OPT

All test cases which uses VM with optimised scheduler policy.

TUNTAP

All test cases which uses TUN and TAP.

AFPKT

All test cases which uses AFPKT.

NETMAP

All test cases which uses Netmap.

MEMIF

All test cases which uses Memif.

6.4.16 Feature tags

IACLDST

iACL destination.

COPWHLIST

COP whitelist.

NAT44

NAT44 configured and tested.

NAT64

NAT44 configured and tested.

ACL

ACL plugin configured and tested.

IACL

ACL plugin configured and tested on input path.

OACL

ACL plugin configured and tested on output path.

ACL_STATELESS

ACL plugin configured and tested in stateless mode (permit action).

ACL_STATEFUL

ACL plugin configured and tested in stateful mode (permit+reflect action).

ACL1

ACL plugin configured and tested with 1 not-hitting ACE.

ACL10

ACL plugin configured and tested with 10 not-hitting ACEs.

ACL50

ACL plugin configured and tested with 50 not-hitting ACEs.

6.4.17 Encryption tags

IPSECSW

Crypto in software.

IPSECHW

Crypto in hardware.

IPSECTRAN

IPSec in transport mode.

IPSECTUN

IPSec in tunnel mode.

6.4.18 Client-workload tags

VM

All test cases which use at least one virtual machine.

LXC

All test cases which use Linux container and LXC utils.

DOCKER

All test cases which use Docker as container manager.

APP

All test cases with specific APP use.

6.4.19 Container orchestration tags

K8S

All test cases which use Kubernetes for orchestration.

SFC_CONTROLLER

All test cases which use ligato/sfc_controller for driving configuration of vpp inside container.

VPP_AGENT

All test cases which use Golang implementation of a control/management plane for VPP

1VSWITCH

VPP running in Docker container acting as VSWITCH.

1VNF

1 VPP running in Docker container acting as VNF work load.

2VNF

2 VPP running in 2 Docker containers acting as VNF work load.

4VNF

4 VPP running in 4 Docker containers acting as VNF work load.

6.4.20 Multi-threading tags

STHREAD

All test cases using single poll mode thread.

MTHREAD

All test cases using more then one poll mode driver thread.

1NUMA

All test cases with packet processing on single socket.

2NUMA

All test cases with packet processing on two sockets.

SMT

All test cases with symmetric Multi-Threading (HyperThreading) enabled.

NOSMT

All test cases with symmetric Multi-Threading (HyperThreading) disabled.

1T1C

1 worker thread pinned to 1 dedicated physical core. 1 receive queue per interface. Main thread pinned to core 0.

2T2C

2 worker threads pinned to 2 dedicated physical cores. 1 receive queue per interface. Main thread pinned to core 0.

4T4C

4 worker threads pinned to 4 dedicated physical cores. 2 receive queues per interface. Main thread pinned to core 0.

6T6C

6 worker threads pinned to 6 dedicated physical cores. 3 receive queues per interface. Main thread pinned to core 0.

8T8C

8 worker threads pinned to 8 dedicated physical cores. 4 receive queues per interface. Main thread pinned to core 0.

6.4.21 Honeycomb tags

HC_FUNC

Honeycomb functional test cases.

HC_NSH

Honeycomb NSH test cases.

HC_PERSIST

Honeycomb persistence test cases.

HC_REST_ONLY

(Exclusion tag) Honeycomb test cases that cannot be run in Netconf mode using ODL client for Restfconf -> Netconf translation.

Bibliography

[lxc] Linux Containers[71]

[lxc-namespace] Resource management: Linux kernel Namespaces and cgroups[72].

[stgraber] LXC 1.0: Blog post series[73].

[lxc-security] Linux Containers Security[74].

[capabilities] **'Linux manual - capabilities - overview of Linux capabilities http://man7.org/linux/man-pages/man7/capabilities.7.html'_.**

[cgroup1] Linux kernel documentation: cgroups[75].

[cgroup2] Linux kernel documentation: Control Group v2[76].

[selinux] SELinux Project Wiki[77].

[lxc-sec-features] LXC 1.0: Security features[78].

[lxc-source] Linux Containers source[79].

[apparmor] Ubuntu AppArmor[80].

[seccomp] SECure COMPuting with filters[81].

[docker] Docker[82].

[k8s-doc] Kubernetes documentation[83].

[ligato] Ligato[84].

[govpp] FD.io goVPP project[85].

[71] https://linuxcontainers.org/
[72] https://www.cs.ucsb.edu/~rich/class/cs293b-cloud/papers/lxc-namespace.pdf
[73] https://stgraber.org/2013/12/20/lxc-1-0-blog-post-series/
[74] https://linuxcontainers.org/lxc/security/
[75] https://www.kernel.org/doc/Documentation/cgroup-v1/cgroups.txt
[76] https://www.kernel.org/doc/Documentation/cgroup-v2.txt
[77] http://selinuxproject.org/page/Main_Page
[78] https://stgraber.org/2014/01/01/lxc-1-0-security-features/
[79] https://github.com/lxc/lxc
[80] https://wiki.ubuntu.com/AppArmor
[81] https://www.kernel.org/doc/Documentation/prctl/seccomp_filter.txt
[82] https://www.docker.com/what-docker
[83] https://kubernetes.io/docs/home/
[84] https://github.com/ligato
[85] https://wiki.fd.io/view/GoVPP

[vpp-agent] Ligato vpp-agent[86].

[image-var] Image parameter is required in initial commit version. There is plan to implement container build class to build Docker/LXC image.

[86] https://github.com/ligato/vpp-agent