PERFORMANCE BENCHMARK

Security Solution Performance Benchmark

Authors

Chih-Hsien Chien
Solution Software Engineer,
Intel Corporation

Eduardo Castro
Solution Software Engineer,
Intel Corporation

1.0 Introduction

Cerner Corporation, Intel Corporation, and Midokura have teamed up to deliver a complete Red Hat Enterprise OpenStack Platform*-based virtual security solution that provides visibility and control against malicious activities within the software-defined data center of Cerner Corporation, securing east-west traffic.

Cerner Corporation is providing its services with private clouds. One of the results of the strict security policy in Cerner is the use of physical devices with intrusion prevention systems (IPS) and intrusion detection systems (IDS), located at the edge of the data centers. Currently, any traffic at Cerner's private cloud must be redirected to remote, physical IPS/IDS devices for packet inspection.

Even if two interconnected virtual machines (VMs) are located on the same physical host, the traffic from the source VM is redirected via the Midokura Enterprise MidoNet* virtual switch to the router and the gateway at the edge of the cloud, to reach IPS/IDS devices. Then it is sent back through the gateway to the cloud and through the router and vSwitch to the target VM. Such behavior is known as a trombone effect and has a significant negative impact on the overall latency.

The proposed solution eliminates the trombone effect and minimizes latency because traffic is inspected locally within the cloud through the use of virtual security functions. In this solution, we do not replace the physical IPS/IDS appliances. These are still used to inspect the north-south and east-west traffic coming from outside the cloud. To support inspection of the east-west traffic within the same cloud, we installed virtual IPS functions at each compute node of the cloud.

The system integration with Red Hat Enterprise OpenStack Platform 7 enables better scalability and automated deployment of virtual functions. The level of security and performance of the cloud solution is at least as good as that of physical IPS appliances.

The Solution Implementation document describes the detailed architecture of the proposed network functions virtualization infrastructure (NFVI) and the integration and configuration of respective software components.

This Performance Benchmark document presents the configuration and results of performance tests conducted on the NFVI. The intended audience is architects and engineers planning to implement and evaluate their own virtualized security architectures.

The intent of this document is to help customers who are interested in implementing security protection mechanisms in Software-Defined Networking (SDN) and Network Functions Virtualization (NFV) domains. It is important to note that the performance results presented herein may differ depending on the hardware configuration and software versions used. Intel does not aim to promote or recommend any specific hardware, software, or supplier mentioned in this document. In addition, Intel does not aim to tie customers to any specific software and hardware stack.

Table of Contents

2.0 System Setup

The solution is using the Intel® Open Network Platform reference architecture deployed on standard high volume servers (SHVS) based on Intel® Xeon® E5-2699 v3 processors running Red Hat Enterprise Linux* 7.1. Table 1 provides the details on configuration of the servers.

All the servers have similar configurations. The two servers based on Intel® Server Board S2600WT2 were dedicated to controller/compute1 and compute2, while Supermicro SuperServer* machines were used in compute3 and Analytic Server.

Table 1. Specification of servers.

SERVER	SPECIFICATION	USAGE
2× Intel® processor-based server	• Intel® Server Board S2600WT2 • 2× Intel® Xeon® processor E5-2699 v3, 2.30 GHz, 45 MB cache, total 36 cores and 72 virtual cores reported in BIOS • 16× 8 GB (total 128 GB) 2133 MHz DDR4 RAM • 2× 1 GbE ports via Intel® Ethernet Controller I350 (rev 01) • 1× 40 GbE QSFP+ via Intel® Ethernet Controller XL710-BM1 (rev 02) • 1 TB HDD, 7200 RPM	• controller/ compute1 • compute2
2× Supermicro SuperServer* 6028U-TR4+	• Supermicro X10DRU-i+ Motherboard • 2× Intel Xeon processor E5-2699 v3, 2.30 GHz, 45 MB cache, total 36 cores and 72 virtual cores reported in BIOS • 8× 16 GB (total 128 GB) 2133 MHz DDR4 RAM • 4× 1 GbE ports via AOC-2UR68-i4G using Intel Ethernet Controller I350 (rev 01) • 1× 40 GbE QSFP+ via Intel Ethernet Controller XL710-BM1 (rev 02) • 2 TB HDD	• compute3 • Analytic Server

The 40 Gigabit Ethernet (GbE) Quad Small Form-factor Pluggable+ (QSFP+) port on the Intel® Ethernet Controller XL710-BM1-based card is used for Data Network connectivity. The External and Management Networks are connected with 1 GbE ports controlled by the Intel® Ethernet Controller i350. The network connectivity and physical topology is presented in Figure 1.

The cloud and the virtual network functions (VNF) orchestration is provided by Red Hat OpenStack Platform* 7. Midokura has supplied the solution with Midokura Enterprise MidoNet 5.02, an enterprise-grade virtual switch, and the Midokura plug-in for Open Security Controller (OSC), acting as SDN controller. This scalable solution is VNF agnostic, and the services are delivered through the OSC 2.5 and the virtual IPS (vIPS) functions. Table 2 lists the software components of the solution.

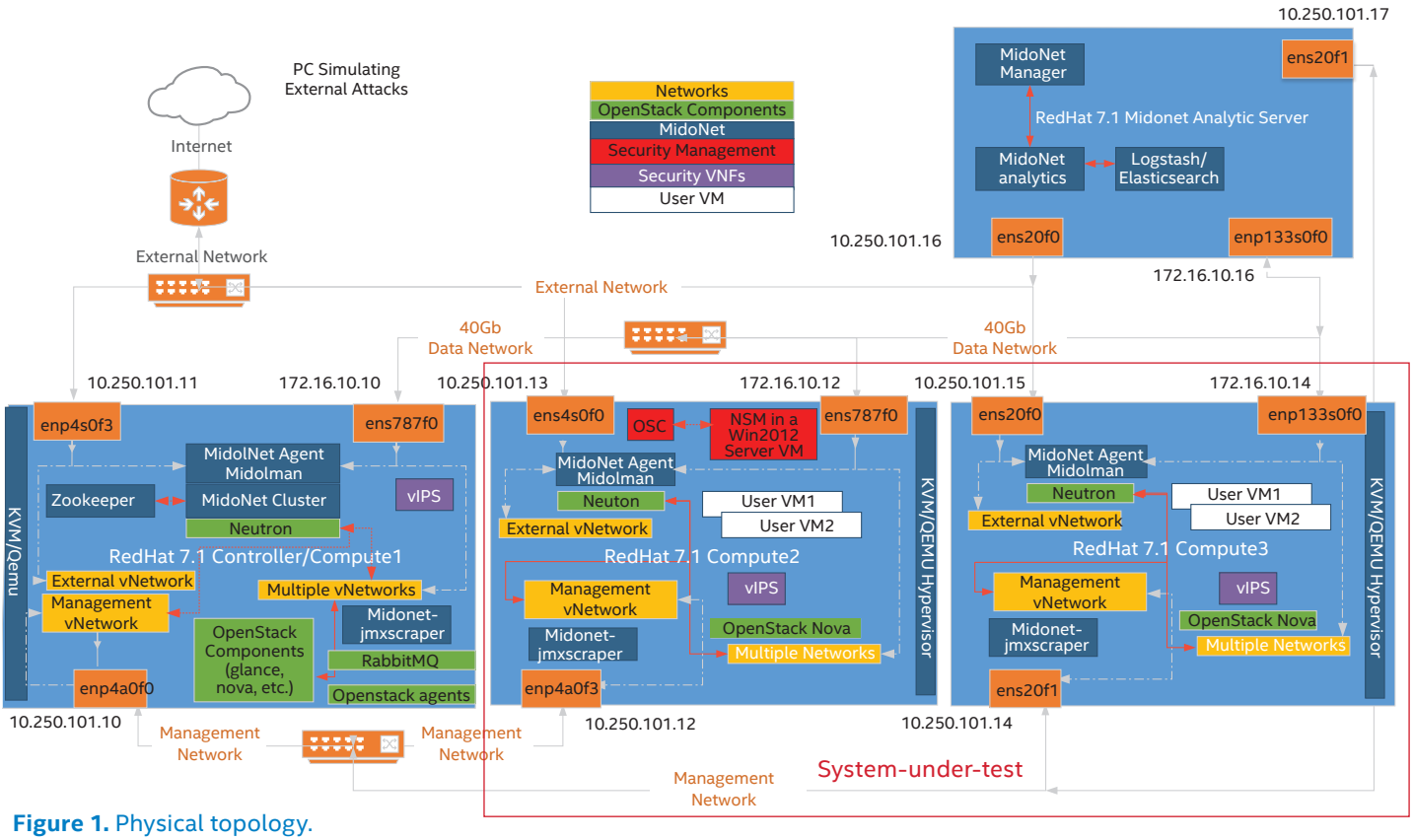

Figure 1. Physical topology.

The Analytic Server provides the network telemetry data and functions supplied by Midokura Enterprise MidoNet and is deployed to enable robust security and network management.

To utilize the capabilities of analytic functions, Midokura and Intel Security implemented a new feature in the SDN controller that uses 6-tuple signature in the API. Widely used 5-tuple-based traffic filtering policy uses the source and destination Internet protocol (IP) addresses and ports, and protocol information. This implementation enhances the 5-tuple signature with information on the flow timestamp of a VM.

OSC and the SDN controller can query the telemetry database on the Midokura Enterprise MidoNet Analytic Server to get all the information about the attacker. This information includes the private IP address of the attacker and the actual name of the VM, even if it is on a different tenant and the attacker traffic is not inspected by the vIPS. With this feature, it is not required to protect all the VMs but only the destination VM. Specifically, implementation modifies the API signature of the SDN controller and the vIPS API that allows you to get information about the source VM and the destination VM of the attack.

Due to the number of services running in the background on the controller/compute1 host that would disrupt the performance, the tests were executed on the compute2 and compute3 nodes.

Table 2. Software components.

FUNCTION	PRODUCT
Operating system	Red Hat Enterprise Linux* 7.1
Hypervisor	qemu-kvm* 2.1.2
Infrastructure orchestration	Red Hat OpenStack Platform* 7
Virtual switching	Midokura Enterprise MidoNet* 5.0.2
SDN controller	Midokura MidoNet plug-in version 1.0034
Security function policy management	Open Security Controller 2.5
Security appliance management	McAfee® Network Security Manager 8.3.7.500.2
Virtual security function with vIPS	McAfee® Network Security Platform virtual sensor 8.1.7.40 (vIPS)
Virtual load balancer	F5 BIG-IP*-12.0.0.0.0.606
Network analytics	Midokura Enterprise MidoNet 5.01

2.1 Tenants

The performance tests were conducted with the existing tenant configuration of the setup. The details on the tenants of the proposed solution can be found in section 4.0 Demo Setup: Cross Tenant Cross Machine Attack of the Security Solution Implementation document. For performance testing, we used Tenant1 and Tenant2 to host server and clients.

2.2 Virtual Machine

The performance benchmark focuses on performance of the VM. Hence it is crucial to define the typical configuration used for the deployment at the customer's lab. The following is the list of customer's requirements for the VM used in our tests.

- Hardware resources:
  - 2× virtual CPUs
  - 4 GB RAM
  - 1× VirtIO device as a virtual network interface card (NIC)
- VM image: Red Hat Enterprise Linux 7.1 Guest image that can be downloaded from the official website of Red Hat at https://access.redhat.com/products/.

Note: For backup purposes, after installation of the OS and the performance tools described in the next section, we recommend creating a snapshot of the VM instance.

2.3 Benchmark Tools

The main focus of benchmarking the Cerner setup was to measure a transmission control protocol (TCP) and a hypertext transfer protocol (HTTP) performance. We used netperf* to measure the performance of TCP, and Apache Benchmark* to measure the performance of HTTP.

Netperf must be installed on both the server and client VMs. The following steps describe the installation process of `netperf` on a single VM.

1. Download netperf from www.netperf.org into the home directory of the guest OS.

```
# cd ~
# wget ftp://www.netperf.org/netperf/
netperf-2.7.0.tar.gz
```

2. Compile and install netperf.

```
# tar xvfz netperf-.2.7.tar.gz
# cd netperf-2.7.0
# ./configure
# make
# make install
```

3. The two binary files, `netperf` and `netserver`, will be installed under `/usr/local/bin/`. On server VM, we run `netserver` with the following command.

```
# /usr/local/bin/netserver
```

4. After `netserver` is started on the server VM, the user can run `netperf` on the client VM to do performance testing. The following sample command executes a `tcp_stream` workload for 60 seconds. Refer to the netperf manpage for more details.

```
# /usr/local/bin/netperf –H <Server IP>
-l 60 -t tcp_stream.
```

Apache Benchmark is a small test tool included in the `apache` package.

1. With Red Hat Enterprise Linux, the installation of `apache` package can be done with the following command. The `apache` package must be installed on both the server and client VMs.

```
# yum install apache
```

2. Once the `apache` package is installed on the server VM, execute the following command to start the Apache server:

```
# apachectl start
```

3. Once Apache server is started, Apache Benchmark can be executed on the client VM with the following sample command:

```
# ab –n 500000 –c 16 http::///<server ip>/
index.html
```

This command will send a total of 500,000 HTTP requests to the server with the specified uniform resource identifier (URI). Option –c denotes that 16 requests will be sent out simultaneously. Option `-k` must be added to test the server with a keep-alive feature enabled. More information on the usage of Apache Benchmark can be found in its manpage.

2.4 Tuning the Test Environment

To start the performance test, ensure the following configuration is done.

1. Enable a jumbo frame feature on a top-of-rack (TOR) switch. An Extreme Networks* switch was used in the performance test setup.

```
# enable jumbo-frame ports all
```

2. Disable the debugger of Midokura Enterprise MidoNet for performance testing because it impacts the performance of the workloads. To disable the debugger, invoke:

```
# echo "agent.loggers.root :INFO" | mn-
conf set -t default
```

To check this setting, invoke:

```
# mn-conf get agent.loggers.root
```

For better performance, we suggest using the MidoNet environment designed for large deployments. To use this environment, invoke the following commands:

```
# mn-conf template-set –h local –agent-
compute-large
```

```
# cp /etc/midolman/midolman-env.
sh.compute.large   /etc/midolman/
midolman-env.sh
```

The above setting for MidoNet should be enabled on all compute nodes. Please make sure to restart the MidoNet agent with following command:

```
# systemctl restart midolman
```

3. The Cerner setup uses a 40 GbE network with Intel Ethernet Controller XL710-BM1 for cross-host traffic. For better performance, we suggest upgrading the firmware to version 5.02 and the driver to version 1.5.16. The upgrade procedure can be found in Appendix A: Intel® Ethernet Controller XL710 Series Firmware and Driver Update. The `ethtool` can be used to check the current firmware and driver version.

```
# ethtool –i enp133s0f0
driver: i40e
version: 1.5.16
firmware-version: 5.02 0x80002285 0.0.0
bus-info: 0000:85:00.0
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
supports-piv-flags: yes
```

After upgrading the firmware and the driver for the Ethernet controller, change the default maximum transmission unit (MTU) size from 1500 to 1600 for the VXLAN tunnel test, and check that the offloading features `rx-checksumming` and `tx-udp_tnl-segmentation` are enabled.

```
# ip link set <XL710 interface> mtu 1600
# ip link set <XL710 interface> down
# ip link set <XL710 interface> up
# ethtool –k <XL710 interface> | grep rx-checksumming
rx-checksumming: on
# ethtool –k <XL710 interface> | grep tx-udp_tnl-segmentation
tx-udp_tnl-segmentation: on
```

3.0 Test Cases and Results

This section describes the test cases defined to understand the network performance that can be achieved with Midokura Enterprise MidoNet in the hosts. The tests were conducted using netperf 2.7 and Apache Benchmark 2.3. The respective performance results are presented at the end of each test case subsection.

3.1 Workload Definition

Although netperf supports various kind of workloads, we focused on the following to measure the performance based on customer's requirements:

- **tcp_stream**: Client continuously sends 16 KB TCP packets to the server with a single TCP connection. This is a throughput test to measure the performance of reception (Rx) of a server.
- **tcp_maerts**: Server continuously sends 16 KB TCP packets to the client with a single TCP connection. Note that `maerts` is `stream` backwards. This is also a throughput test to measure the performance of transmission (Tx) of a server.
- **tcp_rr (TCP Request/Response)**: This workload is used to simulate the behavior of an application such as a webserver over the network. This test can be thought of as a user-space-to-user-space ping with immediate response—it is a synchronous, one transaction at a time, request/response test. Client sends a request to the server and gets a response from the server. One request/response pair is called one transaction. For this workload, all transactions are handled with a single TCP connection. Based on the response size, we can measure performance for both small and large packets. In our tests, we use a 16-byte response for a small packet test and a 4 KB response for a large packet test. In both cases, the request size is fixed at 16 bytes, hence we have two tcp_rr workloads—tcp_rr-16,16 and tcp_rr-16,4k.
- **tcp_crr (TCP Connect/Request/Response)**: This workload is similar to tcp_rr, but with tcp_crr, each transaction will be processed on its own TCP connection. Hence invoking the tcp_crr workload follows the process where at first one TCP connection is created, then a pair of request-response is sent, and finally the connection is closed. Thus this workload can also reflect the connection performance of a system, especially for software-defined networks. Similar to the tcp_rr, we also measure performance for both small and large packets, named tcp_crr-16,16 and tcp_crr-16,4k, respectively.

For Apache Benchmark, we defined four workloads—two workloads for small-size responses and two workloads for large-size responses.

- **request64**: Clients generate HTTP requests and the server sends back 64-byte responses; the keep-alive feature is disabled.
- **request64-k**: Clients generate HTTP requests and the server sends back 64-byte responses; the keep-alive feature is enabled.
- **request4k**: Clients generate HTTP requests and the server sends back 4 KB responses; the keep-alive feature is disabled.
- **request4k-k**: Clients generate HTTP requests and the server sends back 4 KB responses; the keep-alive feature is enabled.

For all Apache Benchmark workloads, we needed to prepare the response in the server with the specified packet size. The keep-alive feature is enabled by default.

For example, in the `request64` workload, we can put one file named `test64.html` of size 64 bytes under the root directory of the Apache HTTP Server (default root directory is `/var/www/httpd`) and let the Apache Benchmark generate the HTTP request with the unified resource locator (URL) `http://<server ip>/test64.html`. Apache Benchmark will then get a 64-byte response from the server.

3.2 Single Host, Single Tenant Network

The workloads for this test case are executed on a single-host machine with a virtual switch from Midokura Enterprise MidoNet. The Perf-Client3 (with the netperf client and Apache Benchmark installed) and Perf-Server-RHEL (with netperf server and Apache HTTP Server installed) are separate VMs located at the same compute2 node, and are connected to the same Tenant2 network.

Figure 2 shows the setup of the test case, while Table 3 and Table 4 provide the commands used for executing netperf client and Apache Benchmark with specific workloads. The test results are given in Table 5.

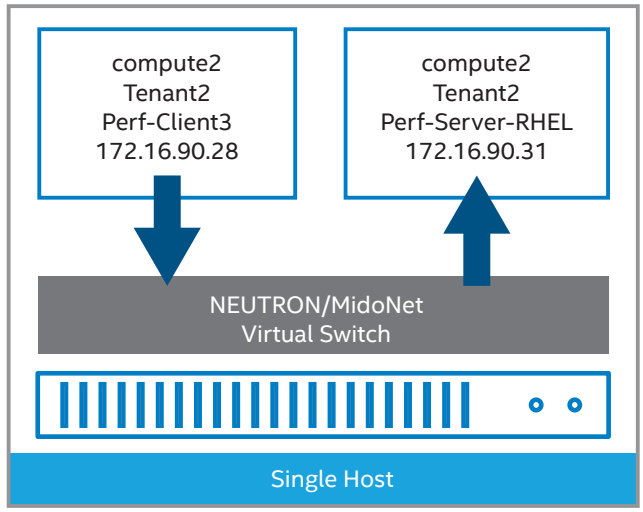

Figure 2. Topology on single host, single tenant.

Table 3. Single host, single tenant—netperf execution.

WORKLOAD	COMMAND
tcp_stream	`netperf –H 172.16.90.31 –l 60 –t tcp_stream –P 0`
tcp_maerts	`netperf –H 172.16.90.31 –l 60 –t tcp_maerts –P 0`
tcp_rr-16,16	`netperf –H 172.16.90.31 –l 60 –t tcp_rr –P 0 -- -b 128 -r 16 16`
tcp_rr-16,4k	`netperf –H 172.16.90.31 –l 60 –t tcp_rr –P 0 -- -b 128 -r 16 4096`
tcp_crr-16,16	`netperf –H 172.16.90.31 –l 60 –t tcp_crr –P 0 -- -b 128 -r 16 16`
tcp_crr-16,4k	`netperf –H 172.16.90.31 –l 60 –t tcp_crr –P 0 -- -b 128 -r 16 4096`

Table 4. Single host, single tenant—Apache Benchmark execution.

WORKLOAD	COMMAND
request64	`ab -n 500000 -c 16 "http://172.10.90.31/test-64.html"`
request64-k	`ab -n 500000 -k -c 16 "http://172.10.90.31/test-64.html"`
request4k	`ab -n 500000 -c 16 "http://172.10.90.31/test-4k.html"`
request4k-k	`ab -n 500000 -k -c 16 "http://172.10.90.31/test-4k.html"`

Table 5. Single host, single tenant—throughput results.

1×VM-1×VM on a single host, single tenant—throughput			
BENCHMARK	**WORKLOAD**	**MEASURE**	**RESULT**
netperf	tcp_stream	Mbps	27,117
	tcp_maerts	Mbps	23,890
	tcp_rr-16,16	transactions/s	1,077,109
	tcp_rr-16,4k	transactions/s	242,724
	tcp_crr-16,16	transactions/s	11,271
	tcp_crr-16,4k	transactions/s	10,368
Apache Benchmark	request64	requests/s	11,940
	request64-k	requests/s	23,369
	request4k	requests/s	12,554
	request4k-k	requests/s	24,004
Test setup	compute2 • Host OS kernel: 3.10.0-229.14.1.el7.x86_64 • Midokura Enterprise MidoNet*: midolman-5.0.2-1.0.hf0.el7.noarch • Guest OS kernel at Client VM: 2.6.32-504.23.4.el6.x86_64 • Guest OS kernel at Server VM: 3.10.0-229.14.1.el7.x86_64		

3.3 Single Host, Separate Tenant Networks

The workloads for this test case are executed on a single-host machine with a virtual switch from Midokura Enterprise MidoNet. The Perf-Client-Tenant1 (with netperf client and Apache Benchmark installed) and Perf-Server2 (with netperf server and Apache HTTP Server installed) are separate VMs located at the same compute3 node but on separate tenants—the VM with test clients is connected to Tenant1, while the VM with test servers is connected to Tenant2.

Figure 3 shows the setup of the test case, while Table 6 and Table 7 provide the commands used for executing netperf client and Apache Benchmark with specific workloads. The test results are given in Table 8.

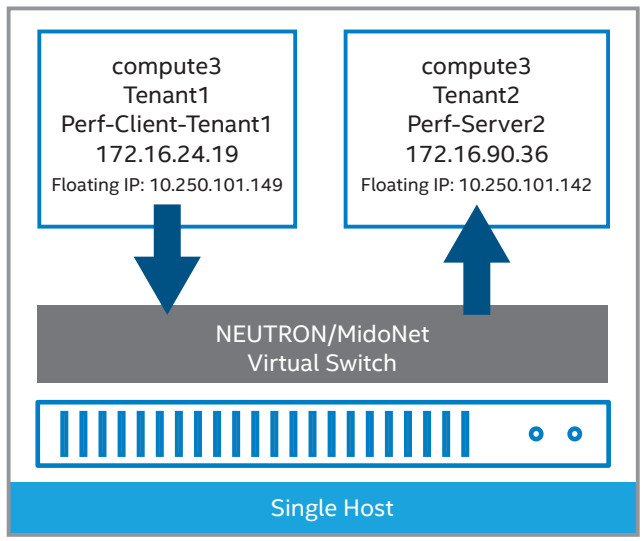

Figure 3. Topology on single host, separate tenants.

Table 6. Single host, separate tenants—netperf execution.

WORKLOAD	COMMAND
tcp_stream	`netperf –H 10.250.101.142 –l 60 –t tcp_stream –P 0`
tcp_maerts	`netperf –H 10.250.101.142 –l 60 –t tcp_maerts –P 0`
tcp_rr-16,16	`netperf –H 10.250.101.142 –l 60 –t tcp_rr –P 0 -- -b 128 -r 16 16`
tcp_rr-16,4k	`netperf –H 10.250.101.142 –l 60 –t tcp_rr –P 0 -- -b 128 -r 16 4096`
tcp_crr-16,16	`netperf –H 10.250.101.142 –l 60 –t tcp_crr –P 0 -- -b 128 -r 16 16`
tcp_crr-16,4k	`netperf –H 10.250.101.142 –l 60 –t tcp_crr –P 0 -- -b 128 -r 16 4096`

Table 7. Single host, separate tenants—Apache Benchmark execution.

WORKLOAD	COMMAND
request64	`ab -n 500000 -c 16 "http://10.250.101.142/test-64.html"`
request64-k	`ab -n 500000 -k -c 16 "http://10.250.101.142/test-64.html"`
request4k	`ab -n 500000 -c 16 "http://10.250.101.142/test-4k.html"`
request4k-k	`ab -n 500000 -k -c 16 "http://10.250.101.142/test-4k.html"`

Table 8. Single host, separate tenants—throughput results.

1×VM-1×VM on a single host, separate tenants—throughput			
BENCHMARK	**WORKLOAD**	**MEASURE**	**RESULT**
netperf	tcp_stream	Mbps	16,087
	tcp_maerts	Mbps	16,201
	tcp_rr-16,16	transactions/s	985,221
	tcp_rr-16,4k	transactions/s	193,710
	tcp_crr-16,16	transactions/s	8,641
	tcp_crr-16,4k	transactions/s	8,780
Apache Benchmark	request64	requests/s	12,535
	request64-k	requests/s	24,183
	request4k	requests/s	12,579
	request4k-k	requests/s	23,892
Test setup	Compute3 • Host OS kernel: 3.10.0-229.14.1.el7.x86_64 • Midokura Enterprise MidoNet*: midolman-5.0.2-1.0.hf0.el7.noarch • Guest OS kernel at Client VM: 2.6.32-504.23.4.el6.x86_64 • Guest OS kernel at Server VM: 3.10.0-229.14.1.el7.x86_64		

3.4 Multiple Clients on a Single Host, Single Tenant Network

The workloads for this test case are executed on a single host machine with a virtual switch from Midokura Enterprise MidoNet. There are four VMs, each one with a netperf client and Apache Benchmark installed, and one VM with a netperf server and Apache HTTP server installed. All VMs are located at the same compute2 node and are connected to the same Tenant2 network.

Figure 4 shows the setup of the test case, while Table 9 and Table 10 provide the commands used for executing the netperf client and Apache Benchmark with specific workloads. The test results are given in Table 11.

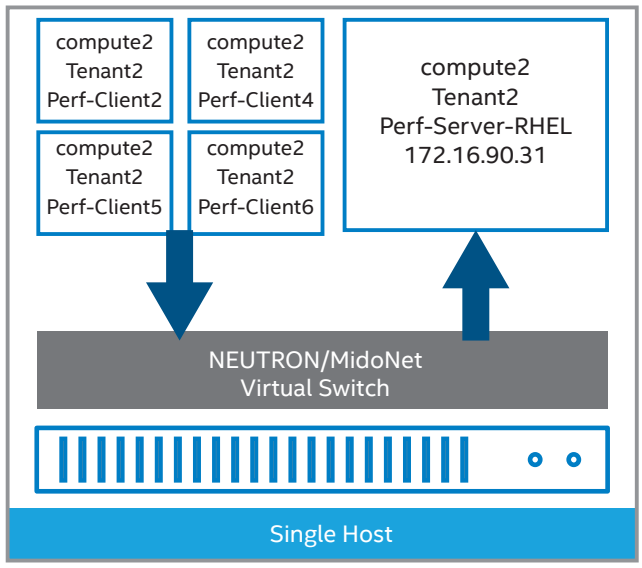

Figure 4. Topology on multiple clients on single host, single tenant.

Table 9. Multiple clients on single host, single tenant—netperf execution.

WORKLOAD	COMMAND
tcp_stream	ssh root@172.16.90.26 netperf -H 172.16.90.31 -l 60 -t tcp_stream -P 0 & ssh root@172.16.90.50 netperf -H 172.16.90.31 -l 60 -t tcp_stream -P 0 & ssh root@172.16.90.51 netperf -H 172.16.90.31 -l 60 -t tcp_stream -P 0 & ssh root@172.16.90.53 netperf -H 172.16.90.31 -l 60 -t tcp_stream -P 0
tcp_maerts	ssh root@172.16.90.26 netperf -H 172.16.90.31 -l 60 -t tcp_maerts -P 0 & ssh root@172.16.90.50 netperf -H 172.16.90.31 -l 60 -t tcp_maerts -P 0 & ssh root@172.16.90.51 netperf -H 172.16.90.31 -l 60 -t tcp_maerts -P 0 & ssh root@172.16.90.53 netperf -H 172.16.90.31 -l 60 -t tcp_maerts -P 0
tcp_rr-16,16	ssh root@172.16.90.26 netperf -H 172.16.90.31 -l 60 -t tcp_rr -P 0 -- -r 16 16 & ssh root@172.16.90.50 netperf -H 172.16.90.31 -l 60 -t tcp_rr -P 0 -- -r 16 16 & ssh root@172.16.90.51 netperf -H 172.16.90.31 -l 60 -t tcp_rr -P 0 -- -r 16 16 & ssh root@172.16.90.53 netperf -H 172.16.90.31 -l 60 -t tcp_rr -P 0 -- -r 16 16
tcp_rr-16,4k	ssh root@172.16.90.26 netperf -H 172.16.90.31 -l 60 -t tcp_rr -P 0 -- -r 16 4096 & ssh root@172.16.90.50 netperf -H 172.16.90.31 -l 60 -t tcp_rr -P 0 -- -r 16 4096 & ssh root@172.16.90.51 netperf -H 172.16.90.31 -l 60 -t tcp_rr -P 0 -- -r 16 4096 & ssh root@172.16.90.53 netperf -H 172.16.90.31 -l 60 -t tcp_rr -P 0 -- -r 16 4096
tcp_crr-16,16	ssh root@172.16.90.26 netperf -H 172.16.90.31 -l 60 -t tcp_crr -P 0 -- -r 16 16 & ssh root@172.16.90.50 netperf -H 172.16.90.31 -l 60 -t tcp_crr -P 0 -- -r 16 16 & ssh root@172.16.90.51 netperf -H 172.16.90.31 -l 60 -t tcp_crr -P 0 -- -r 16 16 & ssh root@172.16.90.53 netperf -H 172.16.90.31 -l 60 -t tcp_crr -P 0 -- -r 16 16
tcp_crr-16,4k	ssh root@172.16.90.26 netperf -H 172.16.90.31 -l 60 -t tcp_crr -P 0 -- -r 16 4096 & ssh root@172.16.90.50 netperf -H 172.16.90.31 -l 60 -t tcp_crr -P 0 -- -r 16 4096 & ssh root@172.16.90.51 netperf -H 172.16.90.31 -l 60 -t tcp_crr -P 0 -- -r 16 4096 & ssh root@172.16.90.53 netperf -H 172.16.90.31 -l 60 -t tcp_crr -P 0 -- -r 16 4096

Table 10. Multiple clients on single host, single tenant—Apache Benchmark execution.

WORKLOAD	COMMAND
request64	ssh root@172.16.90.26 ab -n 500000 -c 16 http://172.16.90.31/test-64.html & ssh root@172.16.90.50 ab -n 500000 -c 16 http://172.16.90.31/test-64.html & ssh root@172.16.90.51 ab -n 500000 -c 16 http://172.16.90.31/test-64.html & ssh root@172.16.90.53 ab -n 500000 -c 16 http://172.16.90.31/test-64.html
request64-k	ssh root@172.16.90.26 ab -n 500000 -c 16 http://172.16.90.31/test-4k.html & ssh root@172.16.90.50 ab -n 500000 -c 16 http://172.16.90.31/test-4k.html & ssh root@172.16.90.51 ab -n 500000 -c 16 http://172.16.90.31/test-4k.html & ssh root@172.16.90.53 ab -n 500000 -c 16 http://172.16.90.31/test-4k.html
request4k	ssh root@172.16.90.26 ab -n 500000 -k -c 16 http://172.16.90.31/test-64.html & ssh root@172.16.90.50 ab -n 500000 -k -c 16 http://172.16.90.31/test-64.html & ssh root@172.16.90.51 ab -n 500000 -k -c 16 http://172.16.90.31/test-64.html & ssh root@172.16.90.53 ab -n 500000 -k -c 16 http://172.16.90.31/test-64.html
request4k-k	ssh root@172.16.90.26 ab -n 500000 -k -c 16 http://172.16.90.31/test-4k.html & ssh root@172.16.90.50 ab -n 500000 -k -c 16 http://172.16.90.31/test-4k.html & ssh root@172.16.90.51 ab -n 500000 -k -c 16 http://172.16.90.31/test-4k.html & ssh root@172.16.90.53 ab -n 500000 -k -c 16 http://172.16.90.31/test-4k.html

Table 11. Multiple clients on single host, single tenant—throughput results.

4×VM-1×VM on a single host, single tenant—throughput			
BENCHMARK	**WORKLOAD**	**MEASURE**	**RESULT**
netperf	tcp_stream	Mbps	1,363
	tcp_maerts	Mbps	4,867
	tcp_rr-16,16	transactions/s	11,222
	tcp_rr-16,4k	transactions/s	9,522
	tcp_crr-16,16	transactions/s	2,525
	tcp_crr-16,4k	transactions/s	2,467
Apache Benchmark	request64	requests/s	2,984
	request64-k	requests/s	5,824
	request4k	requests/s	1,767
	request4k-k	requests/s	6,011
Test setup	Compute2 • Host OS kernel: 3.10.0-229.14.1.el7.x86_64 • Midokura Enterprise MidoNet*: midolman-5.0.2-1.0.hf0.el7.noarch • Guest OS kernel at Client VM: 2.6.32-504.23.4.el6.x86_64 • Guest OS kernel at Server VM: 3.10.0-229.14.1.el7.x86_64		

3.5 Multiple Clients on Single Host, Separate Tenant Networks

The workloads for this test case are executed on a single-host machine with a virtual switch from Midokura Enterprise MidoNet. There are four VMs, each one with netperf client and Apache Benchmark installed, and one VM with a netperf server and Apache HTTP server installed. All VMs are located at the same compute3 node. Contrary to the previous test case, VMs with test clients are connected to Tenant1, while the VMs with test servers are connected to Tenant2.

Figure 5 shows the setup of the test case, while Table 12 and Table 13 provide the commands used for executing the netperf client and Apache Benchmark with specific workloads. The test results are given in Table 14.

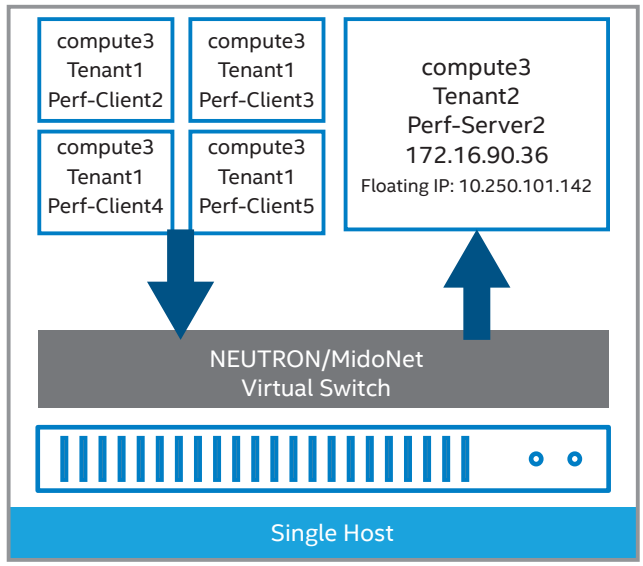

Figure 5. Topology on multiple clients on single host, separate tenants.

Table 12. Multiple clients on single host, separate tenants—netperf execution.

WORKLOAD	COMMAND
tcp_stream	ssh root@172.16.24.12 netperf -H 10.250.101.142 -l 60 -t tcp_stream -P 0 & ssh root@172.16.24.13 netperf -H 10.250.101.142 -l 60 -t tcp_stream -P 0 & ssh root@172.16.24.15 netperf -H 10.250.101.142 -l 60 -t tcp_stream -P 0 & ssh root@172.16.24.15 netperf -H 10.250.101.142 -l 60 -t tcp_stream -P 0
tcp_maerts	ssh root@172.16.24.12 netperf -H 10.250.101.142 -l 60 -t tcp_maerts -P 0 & ssh root@172.16.24.13 netperf -H 10.250.101.142 -l 60 -t tcp_maerts -P 0 & ssh root@172.16.24.14 netperf -H 10.250.101.142 -l 60 -t tcp_maerts -P 0 & ssh root@172.16.24.15 netperf -H 10.250.101.142 -l 60 -t tcp_maerts -P 0
tcp_rr-16,16	ssh root@172.16.24.12 netperf -H 10.250.101.142 -l 60 -t tcp_rr -P 0 -- -r 16 16 & ssh root@172.16.24.13 netperf -H 10.250.101.142 -l 60 -t tcp_rr -P 0 -- -r 16 16 & ssh root@172.16.24.14 netperf -H 10.250.101.142 -l 60 -t tcp_rr -P 0 -- -r 16 16 & ssh root@172.16.24.15 netperf -H 10.250.101.142 -l 60 -t tcp_rr -P 0 -- -r 16 16
tcp_rr-16,4k	ssh root@172.16.24.12 netperf -H 10.250.101.142 -l 60 -t tcp_rr -P 0 -- -r 16 4096 & ssh root@172.16.24.13 netperf -H 10.250.101.142 -l 60 -t tcp_rr -P 0 -- -r 16 4096 & ssh root@172.16.24.14 netperf -H 10.250.101.142 -l 60 -t tcp_rr -P 0 -- -r 16 4096 & ssh root@172.16.24.15 netperf -H 10.250.101.142 -l 60 -t tcp_rr -P 0 -- -r 16 4096
tcp_crr-16,16	ssh root@172.16.24.12 netperf -H 10.250.101.142 -l 60 -t tcp_crr -P 0 -- -r 16 16 & ssh root@172.16.24.13 netperf -H 10.250.101.142 -l 60 -t tcp_crr -P 0 -- -r 16 16 & ssh root@172.16.24.14 netperf -H 10.250.101.142 -l 60 -t tcp_crr -P 0 -- -r 16 16 & ssh root@172.16.24.15 netperf -H 10.250.101.142 -l 60 -t tcp_crr -P 0 -- -r 16 16
tcp_crr-16,4k	ssh root@172.16.24.12 netperf -H 10.250.101.142 -l 60 -t tcp_crr -P 0 -- -r 16 4096 & ssh root@172.16.24.13 netperf -H 10.250.101.142 -l 60 -t tcp_crr -P 0 -- -r 16 4096 & ssh root@172.16.24.14 netperf -H 10.250.101.142 -l 60 -t tcp_crr -P 0 -- -r 16 4096 & ssh root@172.16.24.15 netperf -H 10.250.101.142 -l 60 -t tcp_crr -P 0 -- -r 16 4096

Table 13. Multiple clients on single host, separate tenants—Apache Benchmark execution.

WORKLOAD	COMMAND
request64	ssh root@172.16.24.12 ab -n 500000 -c 16 http://10.250.101.142/test-64.html & ssh root@172.16.24.13 ab -n 500000 -c 16 http://10.250.101.142/test-64.html & ssh root@172.16.24.14 ab -n 500000 -c 16 http://10.250.101.142/test-64.html & ssh root@172.16.24.15 ab -n 500000 -c 16 http://10.250.101.142/test-64.html
request64-k	ssh root@172.16.24.12 ab -n 500000 -c 16 http://10.250.101.142/test-4k.html & ssh root@172.16.24.13 ab -n 500000 -c 16 http://10.250.101.142/test-4k.html & ssh root@172.16.24.14 ab -n 500000 -c 16 http://10.250.101.142/test-4k.html & ssh root@172.16.24.15 ab -n 500000 -c 16 http://10.250.101.142/test-4k.html
request4k	ssh root@172.16.24.12 ab -n 500000 -k -c 16 http://10.250.101.142/test-4k.html & ssh root@172.16.24.13 ab -n 500000 -k -c 16 http://10.250.101.142/test-4k.html & ssh root@172.16.24.14 ab -n 500000 -k -c 16 http://10.250.101.142/test-4k.html & ssh root@172.16.24.15 ab -n 500000 -k -c 16 http://10.250.101.142/test-4k.html
request4k-k	ssh root@172.16.90.26 ab -n 500000 -k -c 16 http://172.16.90.31/test-4k.html & ssh root@172.16.90.50 ab -n 500000 -k -c 16 http://172.16.90.31/test-4k.html & ssh root@172.16.90.51 ab -n 500000 -k -c 16 http://172.16.90.31/test-4k.html & ssh root@172.16.90.53 ab -n 500000 -k -c 16 http://172.16.90.31/test-4k.html

Table 14. Multiple clients on single host, separate tenants—throughput results.

4×VM-1×VM on a single host, separate tenants—throughput			
BENCHMARK	**WORKLOAD**	**MEASURE**	**RESULT**
netperf	tcp_stream	Mbps	6,802
	tcp_maerts	Mbps	5,639
	tcp_rr-16,16	transactions/s	15,691
	tcp_rr-16,4k	transactions/s	14,560
	tcp_crr-16,16	transactions/s	2,444
	tcp_crr-16,4k	transactions/s	3,599
Apache Benchmark	request64	requests/s	3,048
	request64-k	requests/s	3,150
	request4k	requests/s	6,361
	request4k-k	requests/s	6,242
Test setup	Compute3 • Host OS kernel: 3.10.0-229.14.1.el7.x86_64 • Midokura Enterprise MidoNet*: midolman-5.0.2-1.0.hf0.el7.noarch • Guest OS kernel at Client VM: 2.6.32-504.23.4.el6.x86_64 • Guest OS kernel at Server VM: 3.10.0-229.14.1.el7.x86_64		

3.6 Separate Hosts, Single Tenant Network over VXLAN Tunnel

The setup of this test case consists of two host machines. The compute2 node is running a virtual switch from Midokura Enterprise MidoNet, and a single VM with an installation of a netperf client simulating ISO-OSI Layer 3 (L3) traffic and Apache Benchmark simulating ISO-OSI Layer 7 (L7) traffic. The compute3 node is running a virtual switch from Midokura Enterprise MidoNet, and a single VM with an installation of a netperf server and Apache HTTP server. Both VMs are connected to the same tenant network and communicate over the virtual extensible LAN (VXLAN) tunnel.

This is a test case that uses VXLAN to carry L3 and L7 traffic between the VMs located on different physical hosts. Virtual tunnel endpoints (VTEPs) configured on both host machines encapsulate the L3 and L7 traffic into the VXLAN traffic and de-encapsulate the VXLAN traffic received.

Figure 6 presents the setup of the test case, while Table 15 and Table 16 provide the commands used for executing the netperf client and Apache Benchmark with specific workloads. The test results are given in Table 17.

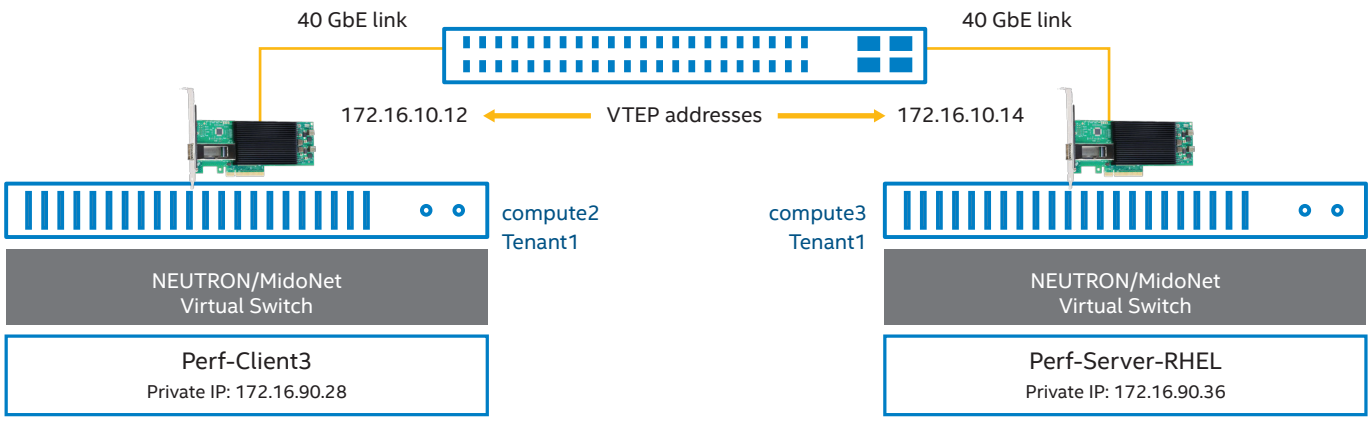

Figure 6. Topology on separate hosts, single tenant over VXLAN tunnel.

Table 15. Separate hosts, single tenant over VXLAN tunnel—Apache Benchmark execution.

WORKLOAD	COMMAND
tcp_stream	`netperf –H 172.16.90.36 –l 60 –t tcp_stream –P 0`
tcp_maerts	`netperf –H 172.16.90.36 –l 60 –t tcp_maerts –P 0`
tcp_rr-16,16	`netperf –H 172.16.90.36 –l 60 –t tcp_rr –P 0 -- -b 128 -r 16 16`
tcp_rr-16,4k	`netperf –H 172.16.90.36 –l 60 –t tcp_rr –P 0 -- -b 128 -r 16 4096`
tcp_crr-16,16	`netperf –H 172.16.90.36 –l 60 –t tcp_crr –P 0 -- -b 128 -r 16 16`
tcp_crr-16,4k	`netperf –H 172.16.90.36 –l 60 –t tcp_crr –P 0 -- -b 128 -r 16 4096`

Table 16. Separate hosts, single tenant over VXLAN tunnel—Apache Benchmark execution.

WORKLOAD	COMMAND
request64	`ab -n 500000 -c 16 "http://172.16.90.36/test-64.html"`
request64-k	`ab -n 500000 -k -c 16 "http://172.16.90.36/test-64.html"`
request4k	`ab -n 500000 -c 16 "http://172.16.90.36/test-4k.html"`
request4k-k	`ab -n 500000 -k -c 16 "http://172.16.90.36/test-4k.html"`

Table 17. Separate hosts, single tenant over VXLAN tunnel—throughput results.

1×VM-1×VM on separate hosts, single tenant over VXLAN tunnel—throughput			
BENCHMARK	**WORKLOAD**	**MEASURE**	**RESULT**
netperf	tcp_stream	Mbps	5,876
	tcp_maerts	Mbps	5,482
	tcp_rr-16,16	transactions/s	799,036
	tcp_rr-16,4k	transactions/s	177,273
	tcp_crr-16,16	transactions/s	6,732
	tcp_crr-16,4k	transactions/s	6,147
Apache Benchmark	request64	requests/s	11,694
	request64-k	requests/s	25,978
	request4k	requests/s	12,561
	request4k-k	requests/s	26,356
Test setup	Compute2 • Host OS kernel: 3.10.0-229.14.1.el7.x86_64 • Midokura Enterprise MidoNet*: midolman-5.0.2-1.0.hf0.el7.noarch • Guest OS kernel: 2.6.32-504.23.4.el6.x86_64 Compute3 • Host OS kernel: 3.10.0-229.14.1.el7.x86_64 • Midokura Enterprise MidoNet: midolman-5.0.2-1.0.hf0.el7.noarch • Guest OS kernel: 3.10.0-229.14.1.el7.x86_64		

3.7 Separate Hosts, Separate Tenant Networks over VXLAN Tunnel

The setup of this test case consists of two host machines. The compute2 node is running a virtual switch from Midokura Enterprise MidoNet, and a single VM with an installation of L3 (netperf) and L7 (Apache Benchmark) traffic test clients on Tenant1. The compute3 node is running a virtual switch from Midokura Enterprise MidoNet, and a single VM with an installation of a netperf server and Apache HTTP server on Tenant2. Both VMs communicate over the VXLAN tunnel.

This is a test case that uses VXLAN to carry L3 and L7 traffic between the VMs located on different physical hosts. VTEPs configured on both host machines encapsulate the L3 and L7 traffic into the VXLAN traffic and de-encapsulate the VXLAN traffic received.

Figure 7 shows the setup of the test case, while Table 18 and Table 19 provide the commands used for executing the netperf client and Apache Benchmark with specific workloads. The test results are given in Table 20.

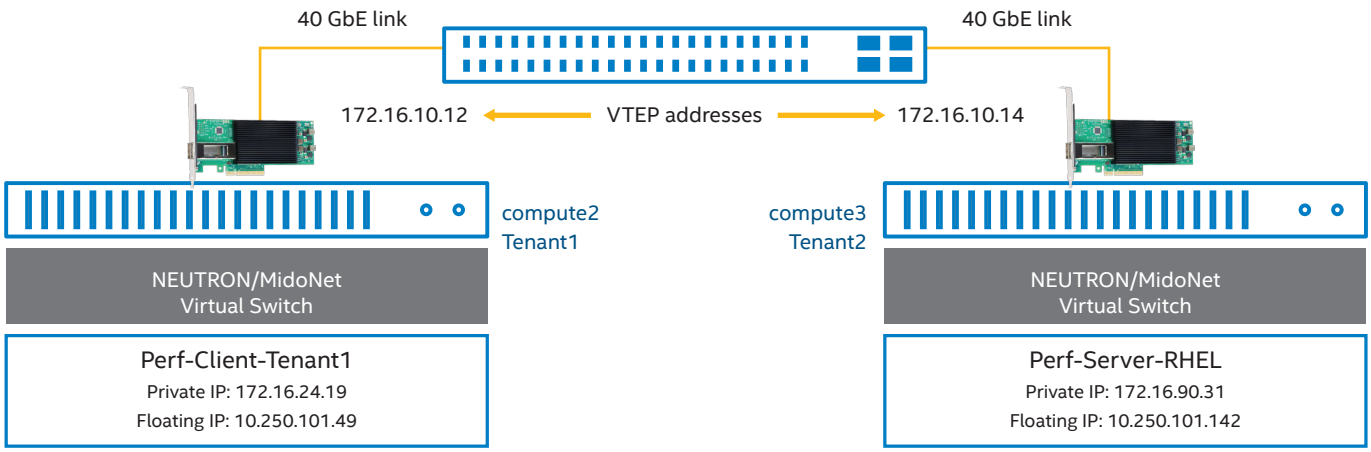

Figure 7. Topology on separate hosts, separate tenants over VXLAN tunnel.

Table 18. Separate hosts, separate tenants over VXLAN tunnel—netperf execution.

WORKLOAD	COMMAND
tcp_stream	`netperf –H 10.250.101.142 –l 60 –t tcp_stream –P 0`
tcp_maerts	`netperf –H 10.250.101.142 –l 60 –t tcp_maerts –P 0`
tcp_rr-16,16	`netperf –H 10.250.101.142 –l 60 –t tcp_rr –P 0 -- -b 128 -r 16 16`
tcp_rr-16,4k	`netperf –H 10.250.101.142 –l 60 –t tcp_rr –P 0 -- -b 128 -r 16 4096`
tcp_crr-16,16	`netperf –H 10.250.101.142 –l 60 –t tcp_crr –P 0 -- -b 128 -r 16 16`
tcp_crr-16,4k	`netperf –H 10.250.101.142 –l 60 –t tcp_crr –P 0 -- -b 128 -r 16 4096`

Table 19. Separate hosts, separate tenants over VXLAN tunnel—Apache Benchmark execution.

WORKLOAD	COMMAND
request64	`ab -n 500000 -c 16 "http://10.250.101.142/test-64.html"`
request64-k	`ab -n 500000 -k -c 16 "http://10.250.101.142/test-64.html"`
request4k	`ab -n 500000 -c 16 "http://10.250.101.142/test-4k.html"`
request4k-k	`ab -n 500000 -k -c 16 "http://10.250.101.142/test-4k.html"`

Table 20. Separate hosts, separate tenants over VXLAN tunnel—throughput results.

1×VM-1×VM on separate hosts, separate tenants—throughput			
BENCHMARK	**WORKLOAD**	**MEASURE**	**RESULT**
netperf	tcp_stream	Mbps	4,372
	tcp_maerts	Mbps	5,699
	tcp_rr-16,16	transactions/s	753,700
	tcp_rr-16,4k	transactions/s	159,484
	tcp_crr-16,16	transactions/s	6,350
	tcp_crr-16,4k	transactions/s	5,446
Apache Benchmark	request64	requests/s	12,638
	request64-k	requests/s	24,019
	request4k	requests/s	12,763
	request4k-k	requests/s	24,914
Test setup	Compute2 • Host OS kernel: 3.10.0-229.14.1.el7.x86_64 • Midokura Enterprise MidoNet*: midolman-5.0.2-1.0.hf0.el7.noarch • Guest OS kernel: 2.6.32-504.23.4.el6.x86_64 Compute3 • Host OS kernel: 3.10.0-229.14.1.el7.x86_64 • Midokura Enterprise MidoNet: midolman-5.0.2-1.0.hf0.el7.noarch • Guest OS kernel: 2.6.32-504.23.4.el6.x86_64		

3.8 Multiple Clients on Single Tenant Network over VXLAN Tunnel

The setup of this test case consists of two host machines. The compute2 node is running a virtual switch from Midokura Enterprise MidoNet and four VMs on Tenant2, each having installed a pair of L3 (netperf) and L7 (Apache Benchmark) traffic test clients. The compute3 node is running a virtual switch from Midokura Enterprise MidoNet, and a single VM with an installation of a netperf server and Apache HTTP server on the same Tenant2. Both VMs communicate over the VXLAN tunnel.

This is a test case that uses VXLAN to carry L3 and L7 traffic between the VMs located on different physical hosts. VTEPs configured on both host machines encapsulate the L3 and L7 traffic into the VXLAN traffic and de-encapsulate the VXLAN traffic received.

Figure 8 shows the setup of the test case, while Table 21 and Table 22 provide the commands used for executing the netperf client and Apache Benchmark with specific workloads. The test results are given in Table 23.

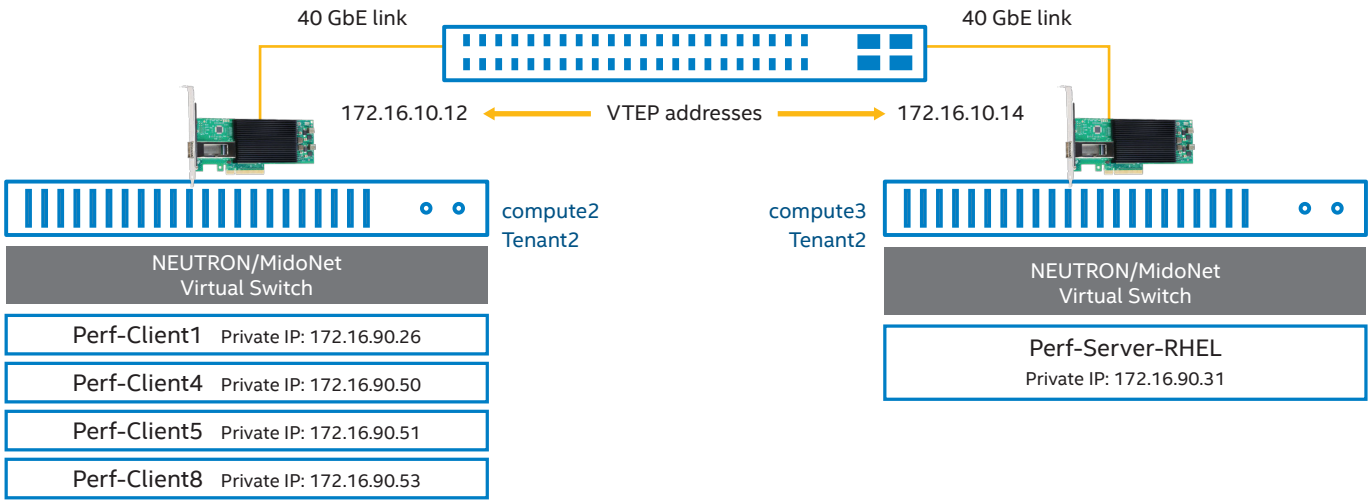

Figure 8. Topology on multiple clients on single tenant over VXLAN tunnel.

Table 21. Multiple clients on single tenant over VXLAN tunnel—netperf execution.

WORKLOAD	COMMAND
tcp_stream	ssh root@172.16.90.26 netperf -H 172.16.90.31 -l 60 -t tcp_stream -P 0 & ssh root@172.16.90.50 netperf -H 172.16.90.31 -l 60 -t tcp_stream -P 0 & ssh root@172.16.90.51 netperf -H 172.16.90.31 -l 60 -t tcp_stream -P 0 & ssh root@172.16.90.53 netperf -H 172.16.90.31 -l 60 -t tcp_stream -P 0
tcp_maerts	ssh root@172.16.90.26 netperf -H 172.16.90.31 -l 60 -t tcp_maerts -P 0 & ssh root@172.16.90.50 netperf -H 172.16.90.31 -l 60 -t tcp_maerts -P 0 & ssh root@172.16.90.51 netperf -H 172.16.90.31 -l 60 -t tcp_maerts -P 0 & ssh root@172.16.90.53 netperf -H 172.16.90.31 -l 60 -t tcp_maerts -P 0
tcp_rr-16,16	ssh root@172.16.90.26 netperf -H 172.16.90.31 -l 60 -t tcp_rr -P 0 -- -r 16 16 & ssh root@172.16.90.50 netperf -H 172.16.90.31 -l 60 -t tcp_rr -P 0 -- -r 16 16 & ssh root@172.16.90.51 netperf -H 172.16.90.31 -l 60 -t tcp_rr -P 0 -- -r 16 16 & ssh root@172.16.90.53 netperf -H 172.16.90.31 -l 60 -t tcp_rr -P 0 -- -r 16 16
tcp_rr-16,4k	ssh root@172.16.90.26 netperf -H 172.16.90.31 -l 60 -t tcp_rr -P 0 -- -r 16 4096 & ssh root@172.16.90.50 netperf -H 172.16.90.31 -l 60 -t tcp_rr -P 0 -- -r 16 4096 & ssh root@172.16.90.51 netperf -H 172.16.90.31 -l 60 -t tcp_rr -P 0 -- -r 16 4096 & ssh root@172.16.90.53 netperf -H 172.16.90.31 -l 60 -t tcp_rr -P 0 -- -r 16 4096
tcp_crr-16,16	ssh root@172.16.90.26 netperf -H 172.16.90.31 -l 60 -t tcp_crr -P 0 -- -r 16 16 & ssh root@172.16.90.50 netperf -H 172.16.90.31 -l 60 -t tcp_crr -P 0 -- -r 16 16 & ssh root@172.16.90.51 netperf -H 172.16.90.31 -l 60 -t tcp_crr -P 0 -- -r 16 16 & ssh root@172.16.90.53 netperf -H 172.16.90.31 -l 60 -t tcp_crr -P 0 -- -r 16 16
tcp_crr-16,4k	ssh root@172.16.90.26 netperf -H 172.16.90.31 -l 60 -t tcp_crr -P 0 -- -r 16 4096 & ssh root@172.16.90.50 netperf -H 172.16.90.31 -l 60 -t tcp_crr -P 0 -- -r 16 4096 & ssh root@172.16.90.51 netperf -H 172.16.90.31 -l 60 -t tcp_crr -P 0 -- -r 16 4096 & ssh root@172.16.90.53 netperf -H 172.16.90.31 -l 60 -t tcp_crr -P 0 -- -r 16 4096

Table 22. Multiple clients on single tenant over VXLAN tunnel—Apache Benchmark execution.

WORKLOAD	COMMAND
request64	ssh root@172.16.90.26 ab -n 500000 -c 16 http://172.16.90.31/test-64.html & ssh root@172.16.90.50 ab -n 500000 -c 16 http://172.16.90.31/test-64.html & ssh root@172.16.90.51 ab -n 500000 -c 16 http://172.16.90.31/test-64.html & ssh root@172.16.90.53 ab -n 500000 -c 16 http://172.16.90.31/test-64.html
request64-k	ssh root@172.16.90.26 ab -n 500000 -c 16 http://172.16.90.31/test-4k.html & ssh root@172.16.90.50 ab -n 500000 -c 16 http://172.16.90.31/test-4k.html & ssh root@172.16.90.51 ab -n 500000 -c 16 http://172.16.90.31/test-4k.html & ssh root@172.16.90.53 ab -n 500000 -c 16 http://172.16.90.31/test-4k.html
request4k	ssh root@172.16.90.26 ab -n 500000 -k -c 16 http://172.16.90.31/test-64.html & ssh root@172.16.90.50 ab -n 500000 -k -c 16 http://172.16.90.31/test-64.html & ssh root@172.16.90.51 ab -n 500000 -k -c 16 http://172.16.90.31/test-64.html & ssh root@172.16.90.53 ab -n 500000 -k -c 16 http://172.16.90.31/test-64.html
request4k-k	ssh root@172.16.90.26 ab -n 500000 -k -c 16 http://172.16.90.31/test-4k.html & ssh root@172.16.90.50 ab -n 500000 -k -c 16 http://172.16.90.31/test-4k.html & ssh root@172.16.90.51 ab -n 500000 -k -c 16 http://172.16.90.31/test-4k.html & ssh root@172.16.90.53 ab -n 500000 -k -c 16 http://172.16.90.31/test-4k.html

Table 23. Multiple clients on single tenant over VXLAN tunnel —throughput results.

4×VM-1×VM on a single tenant over VXLAN—throughput			
BENCHMARK	**WORKLOAD**	**MEASURE**	**RESULT**
netperf	tcp_stream	Mbps	6,521
	tcp_maerts	Mbps	5,931
	tcp_rr-16,16	transactions/s	15,196
	tcp_rr-16,4k	transactions/s	13,896
	tcp_crr-16,16	transactions/s	4,366
	tcp_crr-16,4k	transactions/s	4,310
Apache Benchmark	request64	requests/s	2,878
	request64-k	requests/s	3,050
	request4k	requests/s	5,710
	request4k-k	requests/s	6,073
Test setup	Compute2 • Host OS kernel: 3.10.0-229.14.1.el7.x86_64 • Midokura Enterprise MidoNet*: midolman-5.0.2-1.0.hf0.el7.noarch • Guest OS kernel: 2.6.32-504.23.4.el6.x86_64 Compute3 • Host OS kernel: 3.10.0-229.14.1.el7.x86_64 • Midokura Enterprise MidoNet: midolman-5.0.2-1.0.hf0.el7.noarch • Guest OS kernel: 2.6.32-504.23.4.el6.x86_64		

3.9 Multiple Clients Separate Tenant Networks over VXLAN Tunnel

The setup of this test case consists of two host machines. The compute3 node is running a virtual switch from Midokura Enterprise MidoNet and four VMs on Tenant1, each having installed a pair of L3 (netperf) and L7 (Apache Benchmark) traffic test clients. The compute2 node is running a virtual switch from Midokura Enterprise MidoNet, and a single VM with an installation of a netperf server and Apache HTTP server on Tenant1. Both VMs communicate over the VXLAN tunnel.

This is a test case that uses VXLAN to carry L3 and L7 traffic between the VMs located on different physical hosts. VTEPs configured on both host machines encapsulate the L3 and L7 traffic into the VXLAN traffic and de-encapsulate the VXLAN traffic received.

Figure 9 shows the setup of the test case, while Table 24 and Table 25 provide the commands used for executing the netperf client and Apache Benchmark with specific workloads. The test results are given in Table 26.

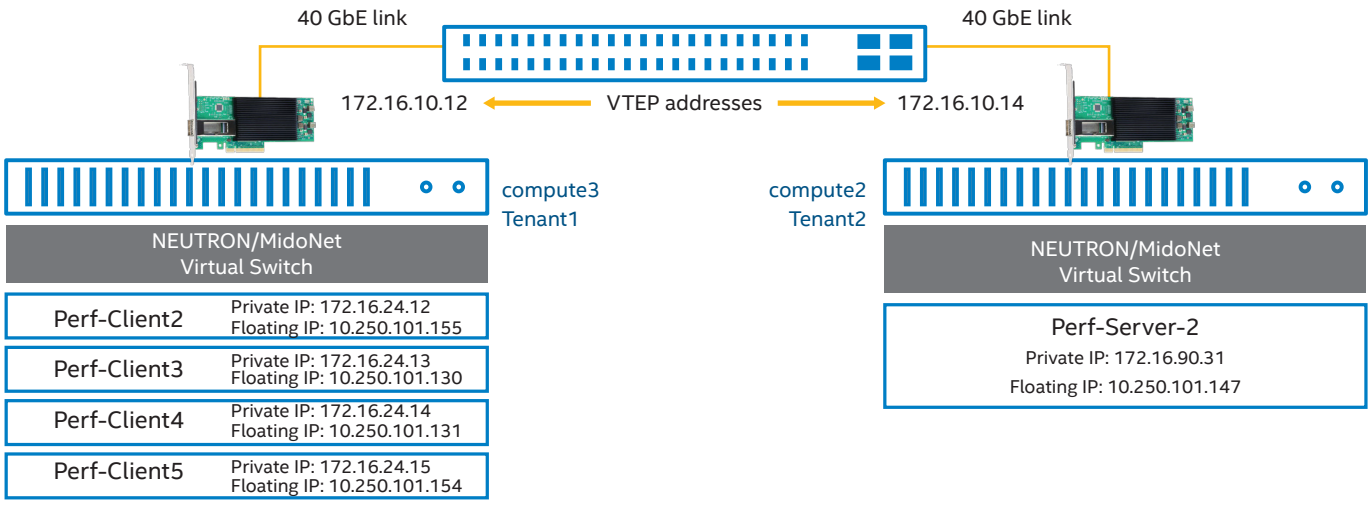

Figure 9. Topology on multiple clients on separate tenants over VXLAN tunnel.

Table 24. Multiple clients on separate tenants over VXLAN tunnel—netperf execution.

WORKLOAD	COMMAND
tcp_stream	ssh root@172.16.24.12 netperf -H 10.250.101.147 -l 60 -t tcp_stream -P 0 & ssh root@172.16.24.13 netperf -H 10.250.101.147 -l 60 -t tcp_stream -P 0 & ssh root@172.16.24.14 netperf -H 10.250.101.147 -l 60 -t tcp_stream -P 0 & ssh root@172.16.24.15 netperf -H 10.250.101.147 -l 60 -t tcp_stream -P 0
tcp_maerts	ssh root@172.16.24.12 netperf -H 10.250.101.147 -l 60 -t tcp_maerts -P 0 & ssh root@172.16.24.13 netperf -H 10.250.101.147 -l 60 -t tcp_maerts -P 0 & ssh root@172.16.24.14 netperf -H 10.250.101.147 -l 60 -t tcp_maerts -P 0 & ssh root@172.16.24.15 netperf -H 10.250.101.147 -l 60 -t tcp_maerts -P 0
tcp_rr-16,16	ssh root@172.16.24.12 netperf -H 10.250.101.147 -l 60 -t tcp_rr -P 0 -- -r 16 16 & ssh root@172.16.24.13 netperf -H 10.250.101.147 -l 60 -t tcp_rr -P 0 -- -r 16 16 & ssh root@172.16.24.14 netperf -H 10.250.101.147 -l 60 -t tcp_rr -P 0 -- -r 16 16 & ssh root@172.16.24.15 netperf -H 10.250.101.147 -l 60 -t tcp_rr -P 0 -- -r 16 16
tcp_rr-16,4k	ssh root@172.16.24.12 netperf -H 10.250.101.147 -l 60 -t tcp_rr -P 0 -- -r 16 4096 & ssh root@172.16.24.13 netperf -H 10.250.101.147 -l 60 -t tcp_rr -P 0 -- -r 16 4096 & ssh root@172.16.24.14 netperf -H 10.250.101.147 -l 60 -t tcp_rr -P 0 -- -r 16 4096 & ssh root@172.16.24.15 netperf -H 10.250.101.147 -l 60 -t tcp_rr -P 0 -- -r 16 4096
tcp_crr-16,16	ssh root@172.16.24.12 netperf -H 10.250.101.147 -l 60 -t tcp_crr -P 0 -- -r 16 16 & ssh root@172.16.24.13 netperf -H 10.250.101.147 -l 60 -t tcp_crr -P 0 -- -r 16 16 & ssh root@172.16.24.14 netperf -H 10.250.101.147 -l 60 -t tcp_crr -P 0 -- -r 16 16 & ssh root@172.16.24.15 netperf -H 10.250.101.147 -l 60 -t tcp_crr -P 0 -- -r 16 16
tcp_crr-16,4k	ssh root@172.16.24.12 netperf -H 10.250.101.147 -l 60 -t tcp_crr -P 0 -- -r 16 4096 & ssh root@172.16.24.13 netperf -H 10.250.101.147 -l 60 -t tcp_crr -P 0 -- -r 16 4096 & ssh root@172.16.24.14 netperf -H 10.250.101.147 -l 60 -t tcp_crr -P 0 -- -r 16 4096 & ssh root@172.16.24.15 netperf -H 10.250.101.147 -l 60 -t tcp_crr -P 0 -- -r 16 4096

Table 25. Multiple clients on separate tenants over VXLAN tunnel—Apache Benchmark execution.

WORKLOAD	COMMAND
request64	ssh root@172.16.24.12 ab -n 500000 -c 16 http://10.250.101.147/test-64.html & ssh root@172.16.24.13 ab -n 500000 -c 16 http://10.250.101.147/test-64.html & ssh root@172.16.24.14 ab -n 500000 -c 16 http://10.250.101.147/test-64.html & ssh root@172.16.24.15 ab -n 500000 -c 16 http://10.250.101.147/test-64.html
request64-k	ssh root@172.16.24.12 ab -n 500000 -c 16 http://10.250.101.147/test-4k.html & ssh root@172.16.24.13 ab -n 500000 -c 16 http://10.250.101.147/test-4k.html & ssh root@172.16.24.14 ab -n 500000 -c 16 http://10.250.101.147/test-4k.html & ssh root@172.16.24.15 ab -n 500000 -c 16 http://10.250.101.147/test-4k.html
request4k	ssh root@172.16.24.12 ab -n 500000 -k -c 16 http://10.250.101.147/test-64.html & ssh root@172.16.24.13 ab -n 500000 -k -c 16 http://10.250.101.147/test-64.html & ssh root@172.16.24.14 ab -n 500000 -k -c 16 http://10.250.101.147/test-64.html & ssh root@172.16.24.15 ab -n 500000 -k -c 16 http://10.250.101.147/test-64.html
request4k-k	ssh root@172.16.24.12 ab -n 500000 -k -c 16 http://10.250.101.147/test-4k.html & ssh root@172.16.24.13 ab -n 500000 -k -c 16 http://10.250.101.147/test-4k.html & ssh root@172.16.24.14 ab -n 500000 -k -c 16 http://10.250.101.147/test-4k.html & ssh root@172.16.24.15 ab -n 500000 -k -c 16 http://10.250.101.147/test-4k.html

Table 26. Multiple clients on separate tenants over VXLAN tunnel—throughput results.

4×VM-1×VM on separate tenants over VXLAN—throughput			
BENCHMARK	**WORKLOAD**	**MEASURE**	**RESULT**
netperf	tcp_stream	Mbps	1,483
	tcp_maerts	Mbps	5,171
	tcp_rr-16,16	transactions/s	8,653
	tcp_rr-16,4k	transactions/s	9,421
	tcp_crr-16,16	transactions/s	1,908
	tcp_crr-16,4k	transactions/s	1,905
Apache Benchmark	request64	requests/s	3,116
	request64-k	requests/s	3,124
	request4k	requests/s	6,042
	request4k-k	requests/s	6,101
Test setup	compute2 • Host OS kernel: 3.10.0-229.14.1.el7.x86_64 • Midokura Enterprise MidoNet*: midolman-5.0.2-1.0.hf0.el7.noarch • Guest OS kernel: 2.6.32-504.23.4.el6.x86_64 compute3 • Host OS kernel: 3.10.0-229.14.1.el7.x86_64 • Midokura Enterprise MidoNet: midolman-5.0.2-1.0.hf0.el7.noarch • Guest OS kernel: 2.6.32-504.23.4.el6.x86_64		

3.10 Latency of Mice and Elephant Flows on a Single Tenant Network

This test case is used to measure the latency of mice and elephant flows. The setup of this test case consists of two host machines. The compute2 node is running a virtual switch from Midokura Enterprise MidoNet and four VMs on Tenant2, each having installed a netperf client. Two VMs are used to simulate mice traffic, sending relatively large amounts of small 16-byte packets. The other two VMs are used to simulate elephant traffic, sending relatively small amounts of large 4 KB packets.

The compute3 node is running a virtual switch from Midokura Enterprise MidoNet and two VMs on the same Tenant2, each having installed a netperf server. One of the servers is dedicated to handling the mice traffic and the other server handles the elephant traffic. All the VMs communicate over the VXLAN tunnel. Separate VXLAN identifiers (VXLAN IDs) are assigned for respective pairs of client-server VMs.

This is a test case that uses VXLAN to carry L3 traffic between the VMs located on different physical hosts. VTEPs configured on both host machines encapsulate the L3 traffic into the VXLAN traffic and de-encapsulate the VXLAN traffic received.

Figure 10 shows the setup of the test case, while Table 27 provides the commands used for executing netperf clients with specific workloads. The test results are given in Table 28.

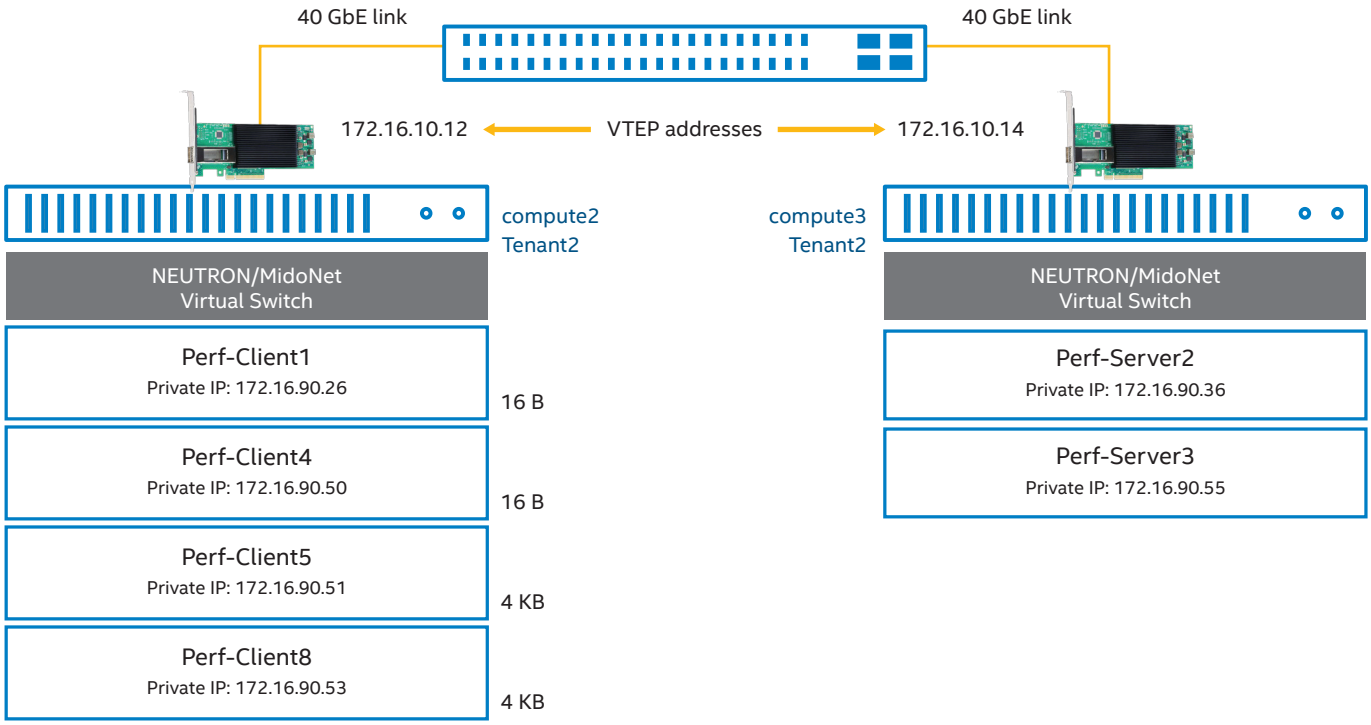

Figure 10. Latency of mice and elephant flows on a single tenant—topology.

Table 27. Mice and elephant flows on a single tenant—netperf execution.

WORKLOAD	COMMAND

tcp_rr-16,16
tcp_rr-16,4k
	`ssh root@172.16.90.26 netperf -H 172.16.90.36 -l 60 -v 2 -t tcp_rr -- -r 16 16 &`
	`ssh root@172.16.90.50 netperf -H 172.16.90.36 -l 60 -v 2 -t tcp_rr -- -r 16 16 &`
	`ssh root@172.16.90.51 netperf -H 172.16.90.55 -l 60 -v 2 -t tcp_rr -- -r 16 4096 &`
	`ssh root@172.16.90.53 netperf -H 172.16.90.55 -l 60 -v 2 -t tcp_rr -- -r 16 4096`

Table 28. Mice and elephant flows on a single tenant—latency results.

2× mice flows (64 B) VMs-1× VM on single tenant over VXLAN 2× elephant flow (4 KB) VMs-1× VM on single tenant over VXLAN				
BENCHMARK	**# OF FLOWS**	**WORKLOAD**	**MEASURE**	**RESULT**
netperf	64 B * 2	tcp_rr-16,16	µs	82
	4 KB * 2	tcp_rr-16,4k	µs	82
Test setup	compute2 • Host OS kernel: 3.10.0-229.14.1.el7.x86_64 • Midokura Enterprise MidoNet*: midolman-5.0.2-1.0.hf0.el7.noarch • Guest OS kernel: 2.6.32-504.23.4.el6.x86_64 compute3 • Host OS kernel: 3.10.0-229.14.1.el7.x86_64 • Midokura Enterprise MidoNet: midolman-5.0.2-1.0.hf0.el7.noarch • Guest OS kernel: 2.6.32-504.23.4.el6.x86_64			

3.11 Latency of Mice and Elephants Flows on Separate Tenant Networks

This test case is used to measure the latency of mice and elephants flows. The setup of this test case consists of two host machines. The compute2 node is running a virtual switch from Midokura Enterprise MidoNet and four VMs on Tenant1, each having installed a netperf client. Two VMs are used to simulate mice traffic, sending relatively large amounts of small 16-byte packets. The other two VMs are used to simulate elephant traffic, sending relatively small amounts of large 4 KB packets.

The compute3 node is running a virtual switch from Midokura Enterprise MidoNet and two VMs, each having installed a netperf server on a different tenant network—Tenant2. One of the servers is dedicated to handling the mice traffic, and the other server handles the elephant traffic. All the VMs communicate over the VXLAN tunnel. Separate VXLAN IDs are assigned for respective pairs of client-server VMs.

This is a test case that uses VXLAN to carry L3 traffic between the VMs located on different physical hosts. VTEPs configured on both host machines encapsulate the L3 traffic into the VXLAN traffic, and de-encapsulate the VXLAN traffic received.

Figure 11 shows the setup of the test case, while Table 29 provides the commands used for executing netperf clients with specific workloads. The test results are given in Table 30.

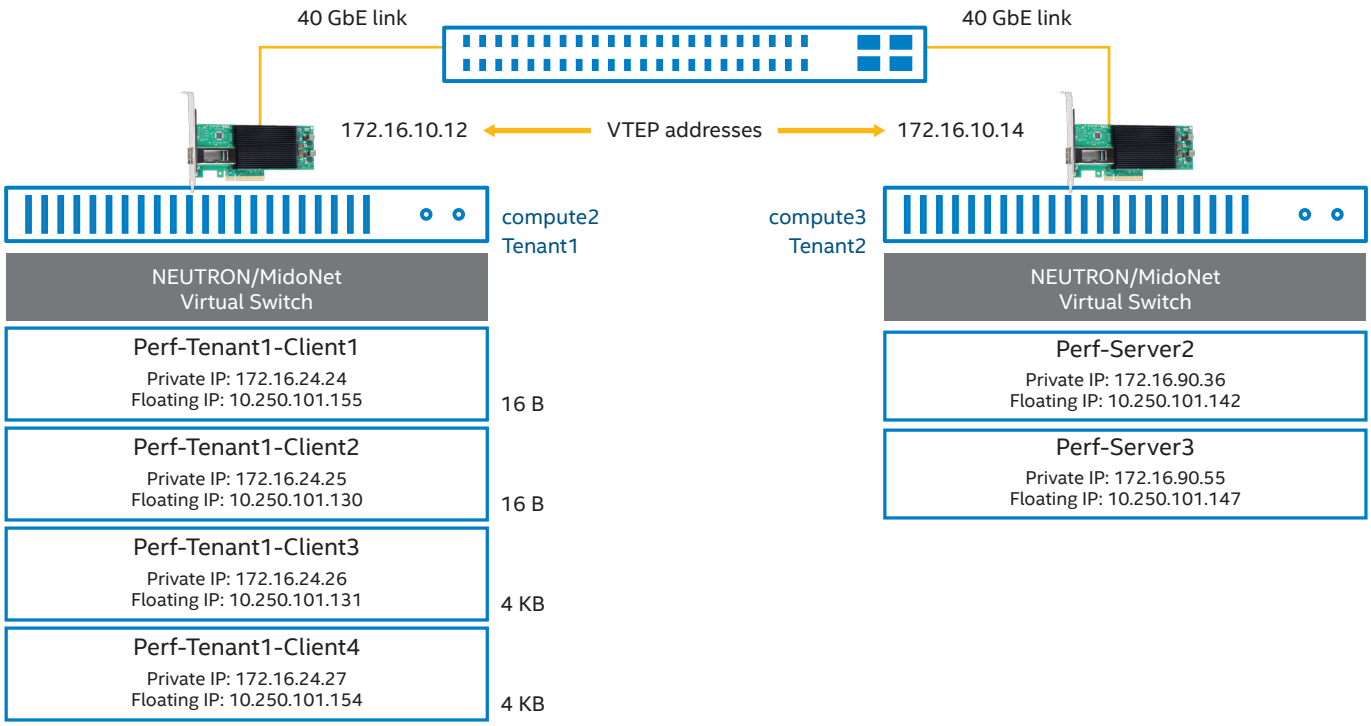

Figure 11. Latency of mice and elephants flow on separate tenants—topology.

Table 29. Mice and elephants flows on separate tenants—netperf execution.

WORKLOAD	COMMAND

tcp_rr-16,16
tcp_rr-16,4k
	`ssh root@172.16.24.24 netperf -H 10.250.101.142 -l 60 -v 2 -t tcp_rr -- -r 16 16 &`
	`ssh root@172.16.24.25 netperf -H 10.250.101.147 -l 60 -v 2 -t tcp_rr -- -r 16 16 &`
	`ssh root@172.16.24.26 netperf -H 10.250.101.142 -l 60 -v 2 -t tcp_rr -- -r 16 4096 &`
	`ssh root@172.16.24.27 netperf -H 10.250.101.147 -l 60 -v 2 -t tcp_rr -- -r 16 4096`

Table 30. Mice and elephant flows on a single tenant—latency results.

2× mice flow (64 B) VMs-1×VM on separate tenant over VXLAN 2× elephants flow (4 KB) VMs-1×VM on separate tenant over VXLAN				
BENCHMARK	**# OF FLOWS**	**WORKLOAD**	**MEASURE**	**RESULT**
netperf	64 B * 2	tcp_rr-16,16	µs	124
	4 KB * 2	tcp_rr-16,4k	µs	125
Test setup	compute2 • Host OS kernel: 3.10.0-229.14.1.el7.x86_64 • Midokura Enterprise MidoNet*: midolman-5.0.2-1.0.hf0.el7.noarch • Guest OS kernel: 2.6.32-504.23.4.el6.x86_64 compute3 • Host OS kernel: 3.10.0-229.14.1.el7.x86_64 • Midokura Enterprise MidoNet: midolman-5.0.2-1.0.hf0.el7.noarch • Guest OS kernel: 2.6.32-504.23.4.el6.x86_64			

Appendix A: Intel® Ethernet Controller XL710 Series Firmware and Driver Update

1. Download the non-volatile memory (NVM) update utility from:

   https://downloadcenter.intel.com/downloads/eula/24769/NVM-Update-Utility-for-Intel-Ethernet-Converged-Network-Adapter-XL710-and-X710-Series?httpDown=https%3A%2F%2Fdownloadmirror.intel.com%2F24769%2Feng%2FNVMUpdatePackage.zip

2. Unzip the downloaded file and extract the `XL710_NVMUpdatePackage_v5_02_Linux.tar.gz` file into `NUMUpdatePackage/XL710` subdirectory.

```
# unzip NVMUpdatePackage.zip
```

```
# tar –zxvf XL710_NVMUpdatePackage_v5_02_Linux.tar.gz
```

3. Provide permissions to `nvmupdate64e`.

```
# cd XL710/Linux_x64
```

```
# chmod +x nvmupdate64e
```

4. Run the file.

```
# /nvmupdate64e –l nvmupdate.log
```

The `nvmupdate64e` update tool will find the target network interface device for update. If the driver is not compatible with the update utility, the adapter will show "Access error" for "Adapter Status" as below.

```
# ./nvmupdate64e –l nvmupdate.log

Intel(R) Ethernet NVM Update Tool
NVMUpdate version 1.26.17.09
Copyright (C) 2013 - 2015 Intel Corporation.

WARNING: To avoid damage to your device, do not stop the update or reboot or power off the system during this update.
Inventory in progress. Please wait [******+...]

Num Description                                 Device-Id B:D    Adapter Status
=== ========================================== ========= ====== ====================
01) Intel(R) I350 Gigabit Network Connecti 8086-1521 01:00  Update not available
02) DH8900CC Series Gigabit Silicon Defaul 8086-0436 131:00 Update not available
03) Intel(R) Ethernet Converged Network Ad 8086-1583 133:00 Access error

Tool execution completed with the following status: Device not found
Press any key to exit.
```

5. In this case, the driver needs to be upgraded. Download the driver file from the link below.

   https://sourceforge.net/projects/e1000/files/i40e%20stable/1.5.16/i40e-1.5.16.tar.gz/download

6. Check the current driver version.

```
# ethtool –i enp133s0f0
driver: i40e
version: 1.0.11-k
firmware-version: f4.40 a1.4 n04.53 e80001f5e
bus-info: 0000:85:00.0
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
supports-piv-flags: no
```

7. Install the `kernel-devel` package based on the kernel version.

```
# uname –r
3.10.0-229.11.1.el7.x86_64
# yum install kernel-devel-3.10.0-229.11.1.el7.x86_64
```

8. Install the `gcc` package.

```
# yum install gcc
```

9. Compile and install the driver.

```
# tar -zxvf i40e-1.5.16.tar.gz
# cd i40e-1.5.16/src
# make
# make install
```

10. Resolve dependence between i40e and vxlan.

```
# echo "softdep i40e pre: vxlan" > /etc/modprobe.d/i40e.conf
```

11. Update the `initramfs` image with the new driver.

```
# dracut –force
```

12. Remove the old driver and install the new driver.

```
# rmmod i40e
# modprobe i40e
```

13. Verify the new driver version.

```
# ethtool –i enp133s0f0
driver: i40e
version: 1.5.16
```

14. After the drive upgrade, run `nvmupdate64e` again.

If adapters show "Update available" for "Adapter Status", select the corresponding adapter number for update, and then press enter to start the update. The update may take a few minutes. After the update is completed, check the new VNM version.

```
# ethtool –i enp133s0f0
driver: i40e
version: 1.5.16
firmware-version: 5.02 0x80002285 0.0.0
bus-info: 0000:85:00.0
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
supports-piv-flags: yes
```

15. Reboot the machine.

Appendix B: References

NAME	REFERENCE
Apache Benchmark	https://httpd.apache.org/docs/2.4/programs/ab.html
McAfee Network Security Manager	http://www.mcafee.com/sg/resources/data-sheets/ds-network-security-manager.pdf
McAfee Network Security Platform virtual sensor	http://www.mcafee.com/us/resources/data-sheets/ds-virtual-network-security-platform.pdf
Midokura Enterprise MidoNet	http://www.midokura.com/midonet-enterprise/
Midokura Enterprise MidoNet (MEM) Quick Start Guide for RHEL 7 / Kilo (OSP)	http://docs.midokura.com/docs/latest-en/quick-start-guide/rhel-7_kilo-osp/content/index.html
Midokura page on GitHub*	https://github.com/midokura
netperf	www.netperf.org
Open Security Controller	http://www.mcafee.com/us/products/open-security-controller.aspx
Red Hat Enterprise Linux 7	https://www.redhat.com/en/resources/red-hat-enterprise-linux-server
Red Hat OpenStack Platform 7	https://access.redhat.com/documentation/en/red-hat-openstack-platform?version=7/

Appendix C: Acronyms and Abbreviations

NAME	REFERENCE
API	Application Programming Interface
BIOS	Basic Input/Output System
CPU	Central Processing Unit
HDD	Hard Disk Drive
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
KB	kilobyte (1 KB = 1024 bytes)
L3	ISO-OSI Layer 3
L7	ISO-OSI Layer 7
LAN	Local Area Network
MTU	Maximum Transmission Unit
NFV	Network Functions Virtualization
NIC	Network Interface Card
NVM	Non-Volatile Memory
OS	Operating System
OSC	Open Security Controller

NAME	REFERENCE
QSFP	Quad Small Form-factor Pluggable
RAM	Random Access Memory
Rx	Reception
SDN	Software-Defined Networking
SHVS	Standard High Volume Servers
TCP	Transmission Control Protocol
TOR	Top-of-Rack
Tx	Transmission
URI	Uniform Resource Identifier
URL	Unified Resource Locator
vIPS	Virtual Intrusion Prevention System
VLAN	Virtual LAN
VM	Virtual Machine
VNF	Virtualized Network Functions
VTEP	Virtual Tunnel End Point
VXLAN	Virtual eXtensible LAN
VXLAN ID	VXLAN Identifier

 1016/MH/MESH/PDF 335200-001US