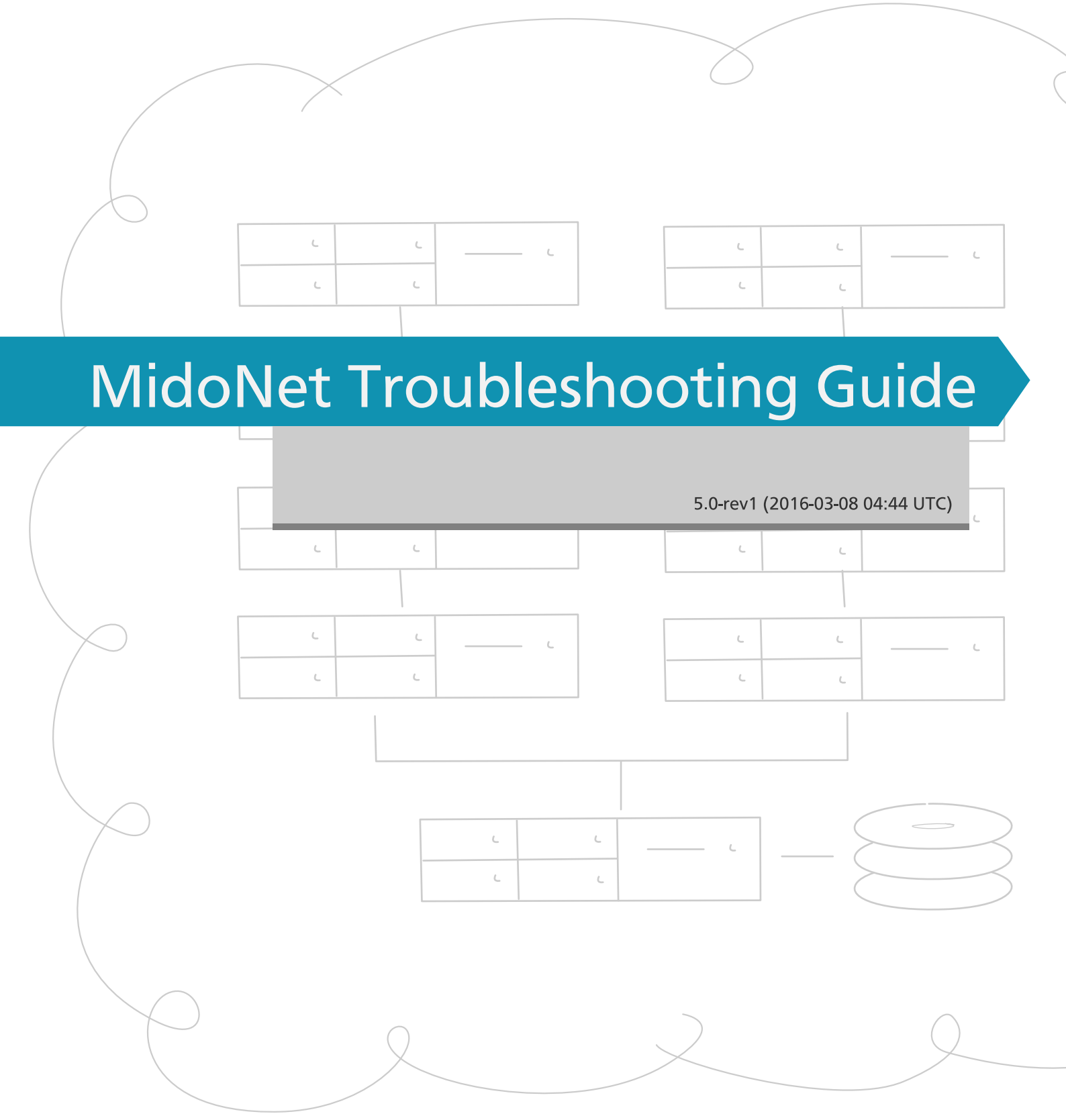

MidoNet Troubleshooting Guide

5.0-rev1 (2016-03-08 04:44 UTC)

midonet

docs.midonet.org

MidoNet Troubleshooting Guide

5.0-rev1 (2016-03-08 04:44 UTC)

MidoNet is a network virtualization software for Infrastructure-as-a-Service (IaaS) clouds.

It decouples your IaaS cloud from your network hardware, creating an intelligent software abstraction layer between your end hosts and your physical network.

This document contains useful information on troubleshooting MidoNet and OpenStack related issues.

Note

Please consult the MidoNet Mailing Lists or Chat if you need assistance.

Table of Contents

Preface

Conventions

The MidoNet documentation uses several typesetting conventions.

Notices

Notices take these forms:

Note

A handy tip or reminder.

Important

Something you must be aware of before proceeding.

Warning

Critical information about the risk of data loss or security issues.

Command prompts

$ prompt

Any user, including the root user, can run commands that are prefixed with the $ prompt.

 # prompt

The root user must run commands that are prefixed with the # prompt. You can also prefix these commands with the **sudo** command, if available, to run them.

1. Overall Approach

Table of Contents

When troubleshooting a MidoNet environment, there are multiple layers to be checked:

- Underlay Network
- Overlay Network
- Virtual Network Topology Simulation
- Virtual Network Topology

Layer	Components
Virtual Network Topology	Neutron, MidoNet NSDB
Virtual Network Topology Simulation	MidoNet Agent
Overlay Network	Tunnel, Datapath
Underlay Network	Physical Environment, Operating System

To rule out possible issues these layers shall be checked from bottom to top.

Underlay Network

The underlay network, i.e. the physical network, should be the first starting point to check for any connectivity issues:

- Hardware / Cabling
- Routing
- Firewall
- Access Control
- Linux Kernel / Open vSwitch
- Time Synchronization

Hardware / Cabling

Ensure that the hardware works properly.

1. **Is the physical link established?**

```
# ethtool eth0
```

```
Settings for eth0:
[...]
   Link detected: yes
```

2. **Is the interface up?**

```
# ip link
[...]
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
 mode DEFAULT group default qlen 1000
    link/ether aa:bb:cc:dd:ee:ff brd ff:ff:ff:ff:ff:ff
```

Routing

Ensure that the routing is configured correctly and check connectivity between hosts via the `ping` command.

```
# netstat -nr
Destination  Gateway      Genmask        Flags  MSS  Window  irtt  Iface
0.0.0.0      192.168.0.1  0.0.0.0        UG     0    0       0     eth0
192.168.0.0  0.0.0.0      255.255.255.0  U      0    0       0     eth0
```

```
# ip route
default via 192.168.0.1 dev eth0  proto static
192.168.0.0/24 dev eth0  proto kernel  scope link  src 192.168.96.100
 metric 1
```

Firewall

Ensure that the firewall is not blocking necessary protocols, hosts or ports.

If unsure, disable the firewall and verify if connectivity issues still persist.

```
# iptables -L
```

Access Control

Ensure that no access control system, such as SELinux or AppArmor, is blocking necessary functionality.

If unsure, disable the ACL system and verify if issues still persist.

Linux Kernel / Open vSwitch

Ensure that the Open vSwitch kernel module is loaded and matches the running Kernel's version.

```
# modinfo openvswitch
filename:       /lib/modules/kernel_version/kernel/net/openvswitch/
openvswitch.ko
license:        GPL
description:    Open vSwitch switching datapath
depends:        libcrc32c,vxlan,gre
intree:         Y
```

```
# lsmod | grep openvswitch
openvswitch            70743  0
vxlan                  37584  1 openvswitch
gre                    13808  1 openvswitch
libcrc32c              12644  2 xfs,openvswitch
```

Time Synchronization

Ensure that the time is synchronized across all nodes.

```
# ntpq -pn
     remote           refid      st t when poll reach   delay   offset
 jitter
==============================================================================
===
*157.7.153.56    133.243.238.164  2 u  114  128  377    4.239    2.713   6.
608
+106.186.114.89  9.22.27.124      3 u   73  128  377    4.845    5.069   4.
802
+157.7.235.92    10.84.87.146     2 u  115  128  377    4.326   14.744   8.
498
+122.215.240.52  133.243.238.164  2 u   45  128  377    4.291    5.400   4.
462
+91.189.94.4     131.188.3.220    2 u   75  128  367  229.564    4.604   6.
896
```

Ensure that correct time zones are configured.

```
# date
Thu Mar 26 13:24:34 JST 2015
```

Overlay Network

The overlay network, i.e. the physical network, should be the first starting point to check for any connectivity issues.

Tunnel Zone

Ensure that the hosts running the MidoNet Agent have been added to the tunnel zone and are alive.

```
# midonet-cli

midonet> list tunnel-zone
tzone tzone0 name tz type vxlan

midonet> tunnel-zone tzone0 list member
zone tzone0 host host0 address 192.168.0.1
zone tzone0 host host1 address 192.168.0.2
zone tzone0 host host2 address 192.168.0.3
zone tzone0 host host3 address 192.168.0.4

midonet> list host
host host0 name host-a alive true
host host1 name host-b alive true
host host2 name host-c alive true
host host3 name host-d alive true
```

Check if packets are transmitted over the tunnel interface, and ensure that there are no errors or dropped packets.

Check the `tnvxlan-overlay` port in case of VXLAN protocol, or the `tngre-overlay` port in case of GRE protocol.

```
# mm-dpctl datapath --show midonet | grep tnvxlan-overlay
Port #x 'tnvxlan-overlay' VXLan
 Stats{rxPackets=24170772, txPackets=45160026, rxBytes=1628556667,
 txBytes=30598767320, rxErrors=0, txErrors=0, rxDropped=0, txDropped=0}
```

```
# mm-dpctl datapath --show midonet | grep tngre-overlay
Port #x "tngre-overlay" Gre Stats{rxPackets=508157678, txPackets=398704120,
 rxBytes=291245619484,
 txBytes=318474308439, rxErrors=0, txErrors=0, rxDropped=0, txDropped=0}
```

MidoNet Datapath

Check the MidoNet Datapath.

```
# mm-dpctl datapath --show midonet
Name: midonet
Index: 10
Supports megaflows: yes
Stats:
  Flows :1340066
  Hits  :1111802509
  Lost  :0
  Misses:17302163
Port #0 "midonet"  Internal  Stats{rxPackets=0, txPackets=0, rxBytes=0,
 txBytes=0, rxErrors=0, txErrors=0, rxDropped=0, txDropped=0}
Port #1 'metadata' Internal Stats{rxPackets=8, txPackets=0, rxBytes=648,
 txBytes=0, rxErrors=0, txErrors=0, rxDropped=0, txDropped=0}
Port #2 "tngre-overlay"  Gre  Stats{rxPackets=508157678,
 txPackets=398704120, rxBytes=291245619484, txBytes=318474308439,
 rxErrors=0, txErrors=0, rxDropped=0, txDropped=0}
Port #3 "tnvxlan-overlay"  VXLan  Stats{rxPackets=0, txPackets=0,
 rxBytes=0, txBytes=0, rxErrors=0, txErrors=0, rxDropped=0, txDropped=0}
Port #4 "tnvxlan-vtep"  VXLan  Stats{rxPackets=0, txPackets=0, rxBytes=0,
 txBytes=0, rxErrors=0, txErrors=0, rxDropped=0, txDropped=0}
Port #5 "tapa0164c42-dd"  NetDev  Stats{rxPackets=389426272,
 txPackets=342761506, rxBytes=1128206548338, txBytes=241007949600,
 rxErrors=0, txErrors=0, rxDropped=0, txDropped=0}
Port #6 "tap19ccc069-f1"  NetDev  Stats{rxPackets=0, txPackets=54640,
 rxBytes=0, txBytes=2347034, rxErrors=0, txErrors=0, rxDropped=0,
 txDropped=0}
Port #7 "tape3055fc6-cc"  NetDev  Stats{rxPackets=21375, txPackets=42911,
 rxBytes=3573207, txBytes=4607633, rxErrors=0, txErrors=0, rxDropped=0,
 txDropped=0}
```

```
# mm-dpctl datapath --dump midonet
1340149 flows
  Flow
    Match: FlowMatch[InputPortNumber=5, EthSrc=0a:b4:2c:fc:86:86,
 EthDst=fa:16:3e:15:6c:cc, EtherType=0800, NetworkSrc=8.8.8.8,
 NetworkDst=172.16.17.18, NetworkProto=udp, NetworkTTL=52, NetworkTOS=0,
 FragmentType=None, SrcPort=53, DstPort=8968]
    Mask:  FlowMask[Priority{0}, InPort{-1},
 Ethernet{src=ff:ff:ff:ff:ff:ff, dst=ff:ff:ff:ff:ff:ff},
 EtherType{0xffffffff}, KeyIPv4{src=255.255.255.255, dst=255.255.255.255,
 proto=-1, tos=-1, ttl=-1, frag=-1}, UDP{src=65535, dst=65535},
 Tunnel{tun_id=0, ipv4_src=0.0.0.0, ipv4_dst=0.0.0.0, tun_flag=0,
 ipv4_tos=0, ipv4_ttl=0}]
    Actions:
      SetKey{Ethernet{src=fa:16:3e:e1:58:2e, dst=fa:16:3e:88:4c:59}}
      SetKey{KeyIPv4{src=8.8.8.8, dst=10.5.0.2, proto=17, tos=0, ttl=50,
 frag=0}}
      SetKey{UDP{src=53, dst=53241}}
      SetKey{Tunnel{tun_id=44, ipv4_src=192.168.123.76,
 ipv4_dst=192.168.123.79, tun_flag=0, ipv4_tos=0, ipv4_ttl=-1}}
      Output{port=3}
    Stats: FlowStats{packets=0, bytes=0}
    TCP flags:
    Last used time: 0
[...]
```

MTU

The MTU of VM instances has to account for the tunnel protocol's overhead to avoid fragmentation in the underlay network.

This adjusted MTU is advertised automatically by MidoNet via DHCP, but may not be applied depending on the VM's operating system being used.

Ensure that the VM's MTU is set accordingly to the underlay's MTU.

Underlay MTU	Tunnel Protocol	Protocol Overhead	VM's MTU
1500 bytes	VXLAN	50 bytes	1450 bytes
1500 bytes	GRE	46 bytes	1455 bytes
9000 bytes	VXLAN	50 bytes	8950 bytes
9000 bytes	GRE	46 bytes	8955 bytes

Topology Simulation

Topology simulation is done by the MidoNet Agent (Midolman), which retrieves the virtual topology data from the Network State Database (NSDB).

Check the `/var/log/midolman/midolman.log` file for errors or warnings.

Ensure that the connection to the NSDB works properly. The NSDB consists of two components, ZooKeeper and Cassandra.

You can verify network accessibility manually by pinging the NSDB hosts and telnetting to the appropriate service ports.

Service	Port
ZooKeeper	2181
Cassandra	9042

```
# telnet nsdb-host 2181
Trying 192.0.2.1...
Connected to nsdb-host.
```

```
# telnet nsdb-host 9042
Trying 192.0.2.1...
Connected to nsdb-host.
```

Virtual Topology

The virtual topology is stored in the Neutron database and MidoNet's Network State Database (NSDB).

Below are some common things to check.

Security Groups

Are the Neutron Security Groups configured to let desired traffic pass?

Check if appropriate rules exist for the protocols (e.g. ICMP, SSH, HTTP) and ports being used.

2. Common Topics

Table of Contents

MidoNet Agent

The MidoNet Agent (Midolman) writes log messages to the `/var/log/midolman/midolman.log` file.

Debugging

During troubleshooting periods, you can increase the logging level via the `mn-conf` utility.

The available log levels are: `DEBUG`, `INFO`, `WARN`, `ERROR`

To check the currently configured level, execute the following command on an Agent host:

```
$ mn-conf get agent.loggers.root
agent.loggers.root = INFO
```

To increase the log level to DEBUG, execute the following command:

```
$ echo "agent.loggers.root : DEBUG" | mn-conf set
```

This change does not require an Agent restart.

MidoNet Cluster

The MidoNet Cluster writes log messages to the `/var/log/midonet-cluster/midonet-cluster.log` file.

Debugging

During troubleshooting periods, you can increase the logging level via the `/etc/midonet-cluster/logback.xml` file.

The available log levels are: `OFF`, `ERROR`, `WARN`, `INFO`, `DEBUG`, `TRACE`

To check the currently configured levels, see the different `logger`s at the end of the file:

```
<logger name="com.sun.jersey" level="WARN"/>
<logger name="org.apache.cassandra" level="INFO"/>
<logger name="org.apache.zookeeper" level="INFO"/>
<logger name="org.eclipse.jetty" level="INFO"/>
```

```
<logger name="org.hibernate" level="WARN"/>
<logger name="org.opendaylight" level="WARN"/>
<logger name="org.reflections" level="INFO"/>

<root level="INFO">
   <appender-ref ref="LOG-FILE" />
</root>
```

To increase the log level to DEBUG, change the logger's levels as follows:

```
<logger name="com.sun.jersey" level="DEBUG"/>
<logger name="org.apache.cassandra" level="DEBUG"/>
<logger name="org.apache.zookeeper" level="DEBUG"/>
<logger name="org.eclipse.jetty" level="DEBUG"/>
<logger name="org.hibernate" level="DEBUG"/>
<logger name="org.opendaylight" level="DEBUG"/>
<logger name="org.reflections" level="DEBUG"/>

<root level="DEBUG">
   <appender-ref ref="LOG-FILE" />
</root>
```

Changing the log level requires a restart of the `midonet-cluster` service.

Border Gateway Protocol (BGP)

This section covers some of the most common BGP related topics.

Find BGP namespaces:

```
# ip netns list
mbgp1_ns
mbgp2_ns
[...]
```

Verify that the Quagga process is running:

```
# ip netns exec mbgp1_ns ps aux | grep [q]uagga
quagga   10614  0.0  0.0 114544  7664 ?        S    04:06   0:00 /usr/
sbin/bgpd --vty_port 2606 --config_file /etc/midolman/quagga/bgpd.conf --
pid_file /var/run/quagga/bgpd.2606.pid --socket /var/run/quagga/zserv1.api
```

Verify that Quagga is listening on both VTYSH and BGP ports:

```
# ip netns exec mbgp1_ns netstat -tulpen
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
     User       Inode      PID/Program name
tcp        0      0 0.0.0.0:2606            0.0.0.0:*               LISTEN
     92         4052614    24029/bgpd
tcp        0      0 0.0.0.0:179             0.0.0.0:*               LISTEN
     92         4052693    24029/bgpd
tcp6       0      0 :::2606                 :::*                    LISTEN
     92         4052615    24029/bgpd
tcp6       0      0 :::179                  :::*                    LISTEN
     92         4052694    24029/bgpd
```

Check the BGP-related interfaces and routing information:

```
# ip netns exec mbgp1_ns ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
```

```
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
5: mbgp1_m: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
 state UP qlen 1000
    link/ether ac:ca:ba:1a:af:24 brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.2/30 scope global mbgp1_m
       valid_lft forever preferred_lft forever
    inet6 fe80::aeca:baff:fe1a:af24/64 scope link
       valid_lft forever preferred_lft forever
7: mbgp1_vtym: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
 state UP qlen 1000
    link/ether c6:89:e2:2b:b0:8e brd ff:ff:ff:ff:ff:ff
    inet 172.23.0.6/30 scope global mbgp1_vtym
       valid_lft forever preferred_lft forever
    inet6 fe80::c489:e2ff:fe2b:b08e/64 scope link
       valid_lft forever preferred_lft forever
```

```
# ip netns exec mbgp1_ns ip route
192.0.2.0/30   dev mbgp1_m     proto kernel  scope link  src 192.0.2.2
172.23.0.4/30  dev mbgp1_vtym  proto kernel  scope link  src 172.23.0.6
```

Ensure that the BGP peers are REACHABLE:

```
# ip netns exec mbgp1_ns ip neigh show
```

Dump the VTYSH interface's traffic:

```
# ip netns exec mbgp1_ns tcpdump -vvv -i mbgp1_vtym
```

Dump ARP, BGP and ICMP traffic on the BGP interface:

```
# ip netns exec mbgp1_ns tcpdump -vvv -i mbgp1_m "icmp or arp or port 179"
```

Dump ARP, BGP and ICMP traffic on the corresponding physical interface:

```
# tcpdump -vvv -i eth1 "icmp or arp or port 179"
```

Launch Quagga's VTY shell:

```
# ip netns exec mbgp1_ns vtysh
```

Inside Quagga's VTY shell, show the running configuration:

```
# show run
Building configuration...

Current configuration:
!
hostname bgpd
log file /var/log/quagga/bgpd.2609.log
!
password zebra_password
!
router bgp 65535
 bgp router-id 192.0.2.2
 network 203.0.113.0/24
 neighbor 192.0.2.1 remote-as 65534
 neighbor 192.0.2.1 timers 5 15
 neighbor 192.0.2.1 timers connect 10
!
line vty
!
end
```

Inside Quagga's VTY shell, show the BGP summary:

```
# show ip bgp summary
BGP router identifier 192.0.2.2, local AS number 65535
RIB entries 19, using 2128 bytes of memory
Peers 1, using 4560 bytes of memory

Neighbor    V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/
PfxRcd
192.0.2.1   4 65534    5512    5523        0    0    0 07:40:09       10
```

Inside Quagga's VTY shell, show BGP routing information:

```
# show ip bgp
BGP table version is 0, local router ID is 192.0.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i -
 internal,
              r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network            Next Hop            Metric LocPrf Weight Path
*> 0.0.0.0            192.0.2.1                0             0 65534 ?
*> 203.0.113.0/24     0.0.0.0                  0         32768 i
*> 192.0.2.0/30       192.0.2.1                0             0 65534 ?

Total number of prefixes 11
```

ZooKeeper

Test if ZooKeeper is running in a non-error state. The server will respond with `imok` if it is running. Otherwise it will not respond at all.

```
$ echo ruok | nc zk-host 2181
imok
```

Lists statistics about performance and connected clients:

```
$ echo stat | nc zk-host 2181
```

Dump the contents of the ZooKeeper database into a pretty-printed text file:

```
$ zkdump -z zk-host:2181 -d -p -o zkdump.txt
```

Dump the contents of the ZooKeeper database into a machine readable JSON file:

```
$ zkdump -z zk-host:2181 -d -o zkdump.json
```

VM Interconnectivity

Scenario

VM1 can not send TCP traffic to VM2.

We want to determine how far the packet is reaching before being lost.

Determine physical compute hosts

To find out on which physical compute hosts these VMs live on, log into Horizon as administrative user and navigate to the instances page.

Find the VMs in the list and note the down the compute hosts, Internal IPs and Floating IPs:

VM1: compute1, 192.168.0.1, 172.16.0.1

VM2: compute2, 192.168.0.2, 172.16.0.2

Determine TAP interfaces

TAP interface names consists of the string "tap", followed by the first 11 characters of the VM's port UUID.

To determine the VM's port UUID, navigate to the VM's network in Horizon and search the port list for the VM's internal IP. Construct the TAP interface name from it like in the following example:

Port UUID: **7aa08012-d0**6c-4c78-aee8-1fff7c063fed

TAP interface: **tap7aa08012-d0**

Examine the traffic on the TAP interfaces

In order to verify if the traffic is being seen on a VM's virtual NIC without logging into the guest host, you can use `tcpdump` on the VM's TAP interface on the compute host.

```
# tcpdump -n -i tap7aa08012-d0
```

Watch packet counters on the TAP interface:

```
# watch -d ip -s link show tap7aa08012-d0
```

3. Tools and Commands

Table of Contents

This section gives an overview of helpful tools and commands.

midonet-cli

The `midonet-cli` command can be run on any host which has the `python-midonet-client` package installed and connectivity to the MidoNet API.

mm-dpctl

The `mm-dpctl` command can be run on any MidoNet Agent node to display the datapath information, such as the current flows.

Available options:

```
# mm-dpctl --help
      --timeout  <arg>   timeout in seconds to wait for an operation to
 complete
                         (default = 3)
      --help             Show help message

Subcommand: datapath - datapath operations (create, delete, list, dump)
  -a, --add  <arg>      add a new datapath
  -D, --delete  <arg>   delete a datapath
  -d, --dump  <arg>     show all the flows installed for a given datapath
  -l, --list            list all the installed datapaths
  -s, --show  <arg>     show all the information related to a given
 datapath
      --help            Show help message
Subcommand: interface - interface operations (add, delete)
  -a, --add  <arg>      add an interface to a datapath
  -d, --delete  <arg>   delete an interface from a datapath
      --help            Show help message

 trailing arguments:
  datapath (required)   the datapath where to create/delete a flow
Subcommand: flows - flow operations (create, delete)
      --arp               src-ip and dst-ip belong to an ARP
  -d, --datapath  <arg>   the datapath where to create/delete a flow
  -D, --delete            delete the flow
      --dst-ip  <arg>     destination IP/ARP address
      --dst-mac  <arg>    destination MAC address
      --dst-port  <arg>   transport destination port or ICMP code
  -i, --input  <arg>      input interface (adds it to datapath)
  -o, --output  <arg>     output interface (adds it to datapath)
      --proto  <arg>      IP protocol (TCP, UDP, or ICMP)
      --reply             tells whether the ARP is a reply
      --src-ip  <arg>     source IP/ARP address
```

```
      --src-mac  <arg>    source MAC address
      --src-port  <arg>   transport source port or ICMP type
      --help              Show help message
```

Examples:

```
# mm-dpctl datapath --show midonet # shows datapath and interfaces
# mm-dpctl datapath --dump midonet # dumps current flows
```

mm-trace

The `mm-trace` command allows the MidolMan Agent to capture a particular traffic flow and to log each stage of the simulation.

It's settings are not persistent across MidolMan restarts.

Outputs are written to the `/var/log/midolman/mm-trace.log` file.

Available options:

```
$ mm-trace --help
  -h, --host  <arg>   Host (default = localhost)
  -p, --port  <arg>   JMX port (default = 7200)
      --help          Show help message

Subcommand: add - add a packet tracing match
  -d, --debug               logs at debug level
      --dst-port  <arg>     match on TCP/UDP destination port
      --ethertype  <arg>    match on ethertype
      --ip-dst  <arg>       match on ip destination address
      --ip-protocol  <arg>  match on ip protocol field
      --ip-src  <arg>       match on ip source address
  -l, --limit  <arg>        number of packets to match before disabling
 this trace
      --mac-dst  <arg>      match on destination MAC address
      --mac-src  <arg>      match on source MAC address
      --src-port  <arg>     match on TCP/UDP source port
  -t, --trace               logs at trace level
      --help                Show help message
Subcommand: remove - remove a packet tracing match
  -d, --debug               logs at debug level
      --dst-port  <arg>     match on TCP/UDP destination port
      --ethertype  <arg>    match on ethertype
      --ip-dst  <arg>       match on ip destination address
      --ip-protocol  <arg>  match on ip protocol field
      --ip-src  <arg>       match on ip source address
  -l, --limit  <arg>        number of packets to match before disabling
 this trace
      --mac-dst  <arg>      match on destination MAC address
      --mac-src  <arg>      match on source MAC address
      --src-port  <arg>     match on TCP/UDP source port
  -t, --trace               logs at trace level
      --help                Show help message
Subcommand: flush - clear the list of tracing matches
  -D, --dead-only   flush expired tracers only
      --help        Show help message
Subcommand: list - list all active tracing matches
  -L, --live-only   list active tracers only
      --help        Show help message
```

Examples:

```
$ mm-trace list
```

```
$ mm-trace add --debug --ip-dst 192.0.2.1
$ mm-trace add --trace --ip-src 192.0.2.1 --dst-port 80
$ mm-trace list
tracer: --debug --ip-dst 192.0.2.1
tracer: --trace --ip-src 192.0.2.1 --dst-port 80
$ mm-trace remove --trace --ip-src 192.0.2.1 --dst-port 80
Removed 1 tracer(s)
$ mm-trace flush
Removed 1 tracer(s)
```

ip

The `ip` command can be used to show / manipulate routing, devices, policy routing and tunnels.

See the man page for detailed information: http://linux.die.net/man/8/ip

List interfaces:

```
# ip link show
```

List namespaces:

```
# ip netns list
```

List interfaces within a namespace:

```
# ip netns exec namespace ip link show
```

4. Directories and Files

Table of Contents

This section gives an overview of frequently used configuration and log files.

Note that file names and paths may slightly differ depending on the used operating system and OpenStack distribution.

Cassandra

File	Type
/etc/cassandra/conf/cassandra.yaml (RHEL) /etc/cassandra/cassandra.yaml (Ubuntu)	CONF
/var/log/cassandra/cassandra.log (RHEL) /var/log/cassandra/system.log (Ubuntu)	LOG

MidoNet Agent

File	Type
/etc/midolman/midolman.conf	CONF
/etc/midolman/midolman-env.sh	CONF
/var/log/midolman/midolman.event.log	LOG
/var/log/midolman/midolman.log	LOG
/var/log/midolman/mm-trace.log	LOG
/var/log/midolman/upstart-stderr.log	LOG

MidoNet Cluster

File	Type
/var/log/midonet-cluster/midonet-cluster.log	LOG
/var/log/midonet-cluster/upstart-stderr.log	LOG

Quagga (BGPD)

File	Type
/etc/midolman/quagga/*	CONF
/var/log/midolman/quagga.*xxxx*/*	LOG
/var/log/quagga/bgpd.log	LOG

ZooKeeper

File	Type
/etc/zookeeper/zoo.cfg (RHEL) /etc/zookeeper/conf/zoo.cfg (Ubuntu)	CONF
/var/log/zookeeper/zookeeper.out (RHEL) /var/log/zookeeper/zookeeper.log (Ubuntu)	LOG

5. Processes

This section gives an overview of common processes.

Note that exact names and paths may differ depending on the used operating system and OpenStack distribution.

Program	Process
Cassandra	java *[...]* org.apache.cassandra.service.CassandraDaemon
MidoNet Agent	java *[...]* org.midonet.midolman.Midolman *[...]*
MidoNet Agent (Watchdog)	/usr/bin/wdog
MidoNet Cluster	java *[...]* org.midonet.cluster.ClusterNode *[...]*
ZooKeeper	java *[...]* org.apache.zookeeper.server.quorum.QuorumPeerMain *[...]*