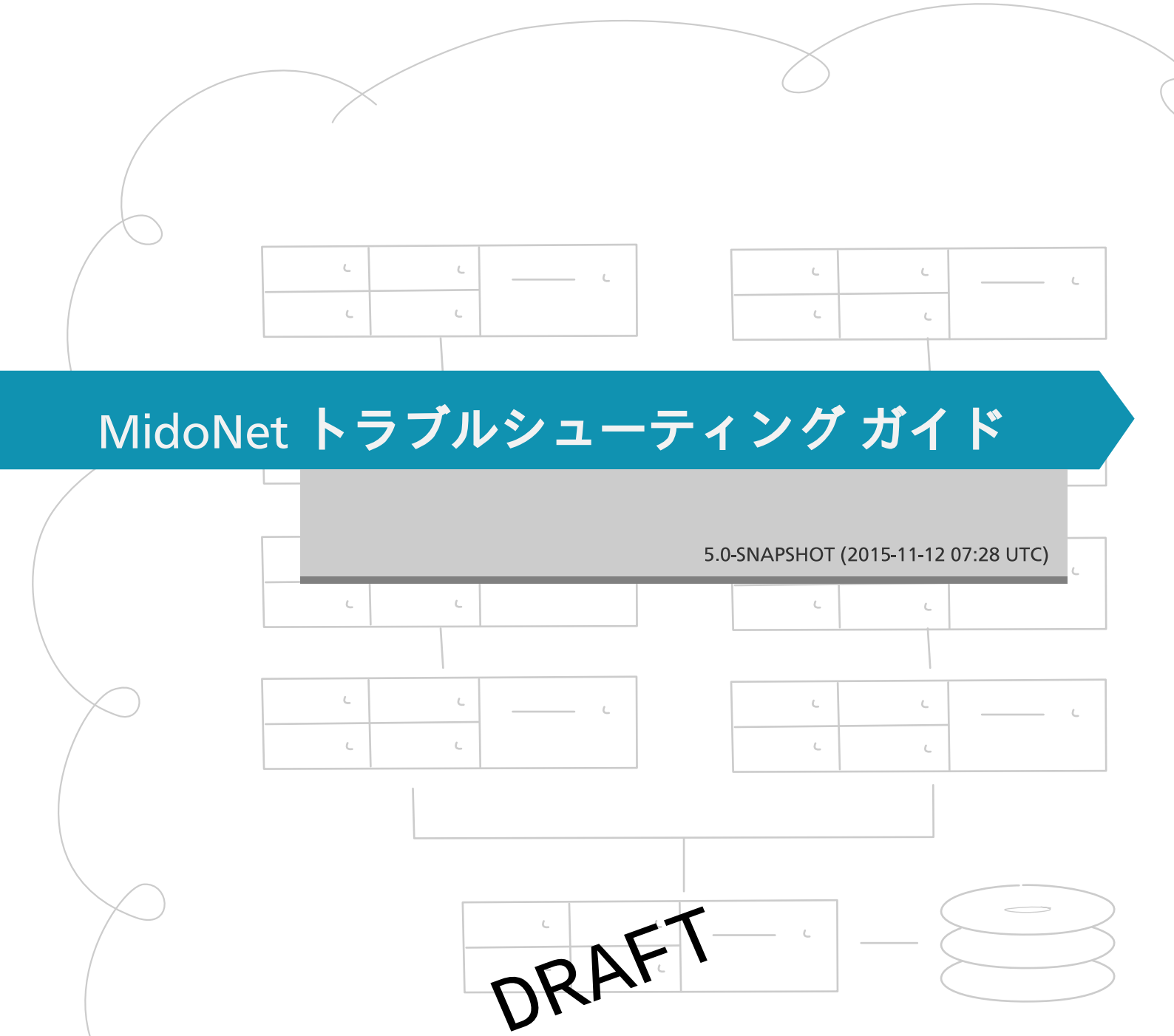
MidoNet トラブルシューティング ガイド

5.0-SNAPSHOT (2015-11-12 07:28 UTC)



midonet

docs.midonet.org

MidoNet トラブルシューティング ガイド

5.0-SNAPSHOT (2015-11-12 07:28 UTC)

概要

MidoNetは、Infrastructure-as-a-Service (IaaS)のためのネットワーク仮想化ソフトウェアです。

これにより、ネットワークハードウェアとIaaSクラウドを切り離すことができ、ホストと物理ネットワークの間に、インテリジェントなソフトウェア抽象レイヤーを作成することができます。

このドキュメントは、MidoNetやOpenStackのトラブルシューティングに関する有用な情報が含まれています。

注意

この文書はドラフトです。それは、関連する情報が欠落しているか、テストされていない情報が含まれていることができる。 ご自身の責任でそれを使用してください。

注記

援助を必要とする場合は、 MidoNetメーリングリストやチャット までご連絡ください。

目次

はじめに

表記規則

MidoNet のドキュメントは、いくつかの植字の表記方法を採用しています。

注意

注意には以下の種類があります。

注記

簡単なヒントや備忘録です。

重要

続行する前に注意する必要があるものです。

警告

データ損失やセキュリティ問題のリスクに関する致命的な情報です。

コマンドプロンプト

$ プロンプト

root ユーザーを含むすべてのユーザーが、$ プロンプトから始まるコマンドを実行できます。

プロンプト

root ユーザーは、# プロンプトから始まるコマンドを実行する必要があります。利用可能ならば、 これらを実行するために、sudo コマンドを使用できます。

第1章 全体アプローチ

目次

MidoNetの環境でトラブルシューティングをする時は、複数のレイヤーをチェックする必要があります。

- アンダーレイネットワーク
- オーバーレイネットワーク
- 仮想ネットワークトポロジーシュミレーション
- 仮想ネットワークトポロジー

レイヤー	コンポーネント
仮想ネットワークトポロジー	Neutron, MidoNet NSDB
仮想ネットワークトポロジーシュミレーション	MidoNet エージェント
オーバーレイネットワ	トンネル、データパス
アンダーレイネットワー	物理環境, オペレーティングシステム

潜在的な問題を避けるために、これらのレイヤーはボトムからトップにかけてチェックしてください。

アンダーレイネットワーク

アンダーレイネットワーク、つまり物理ネットワークは、接続に関する問題があった時に一番最初に確認するポイントです。

- ハードウェア / ケーブル
- ルーティング
- ファイヤーウォール
- アクセスコントロール
- リナックスカーネル / オープン vSwitch
- 時間の同期化

ハードウェア/ ケーブル

ハードウェアが正常に機能しているか確認してください。

1. 物理リンクは確立されていますか？

```
# ethtool eth0
Settings for eth0:
```

```
[...]
   Link detected: yes
```

2. インターフェースは立ち上がっていますか？

```
# 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
```

ルーティング

ルーティングが適切に設定されているか確認してください。そして、ping コマンドを使ってホスト間の接続性を確認してください。

```
# 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
```

ファイヤーウォール

ファイヤーウォールが必要なプロトコル、ホスト、ポートをブロックしていないか確認してください。

もし確信の無い場合は、ファイヤーウォールを解除して、接続に関する問題が残っていないか検証してください。

```
# iptables -L
```

Access Control

SELinux もしくは AppArmorなどのアクセスコントロールシステムが、必要な機能をブロックしていないかを確認してください。

もし確信が無い場合は、ACLシステムを解除して、まだ問題があるかどうかを検証してください。

Linuxカーネル / オープンvSwitch

オープンvSwitchカーネルモジュールがロードされていて、走っているカーネルのバージョンと合っているか確認してください。

```
# 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
```

時間の同期化

全てのノードにおいて、時間が同期しているか確認してください。

```
# 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
```

正しいタイムゾーンが設定されているか確認ください。

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

オーバーレイネットワーク

オーバーレイネットワーク、つまり物理ネットワークは、接続に関する問題があった時にまず一番最初にチェックするポイントです。

トンネルゾーン

MidoNetエージェントを走らせているホストが、トンネルゾーンに追加されていて、稼働していることを確認してください。

```
# 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
```

パケットがトンネルインターフェース上を伝送しているかどうかをチェックして、エラーが無いこと、パケットがドロップしていないことを確認してください。 GREプロトコルの場合は、`tngre-overlay`を確認してください。VXLANプロトコルの場合は、`tnvxlan-overlay`を確認してください。

```
# mm-dpctl --show-dp midonet | grep overlay
Port #1 "tngre-overlay"  Gre  Stats{rxPackets=508157678, txPackets=398704120,
 rxBytes=291245619484, txBytes=318474308439, rxErrors=0, txErrors=0, rxDropped=0,
 txDropped=0}
Port #2 "tnvxlan-overlay"  VXLan  Stats{rxPackets=0, txPackets=0, rxBytes=0, txBytes=0,
 rxErrors=0, txErrors=0, rxDropped=0, txDropped=0}
```

MidoNet データパス

MidoNetデータパスを確認してください。

```
# mm-dpctl --show-dp midonet
```

```
Datapath name   : midonet
Datapath index : 11
Datapath 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 "tngre-overlay"  Gre  Stats{rxPackets=508157678, txPackets=398704120,
 rxBytes=291245619484, txBytes=318474308439, rxErrors=0, txErrors=0, rxDropped=0,
 txDropped=0}
Port #2 "tnvxlan-overlay"  VXLan  Stats{rxPackets=0, txPackets=0, rxBytes=0, txBytes=0,
 rxErrors=0, txErrors=0, rxDropped=0, txDropped=0}
Port #3 "tnvxlan-vtep"  VXLan  Stats{rxPackets=0, txPackets=0, rxBytes=0, txBytes=0,
 rxErrors=0, txErrors=0, rxDropped=0, txDropped=0}
Port #4 "tapa0164c42-dd"  NetDev  Stats{rxPackets=389426272, txPackets=342761506,
 rxBytes=1128206548338, txBytes=241007949600, rxErrors=0, txErrors=0, rxDropped=0,
 txDropped=0}
Port #5 "tap19ccc069-f1"  NetDev  Stats{rxPackets=0, txPackets=54640, rxBytes=0,
 txBytes=2347034, rxErrors=0, txErrors=0, rxDropped=0, txDropped=0}
Port #6 "tape3055fc6-cc"  NetDev  Stats{rxPackets=21375, txPackets=42911, rxBytes=3573207,
 txBytes=4607633, rxErrors=0, txErrors=0, rxDropped=0, txDropped=0}
```

```
# mm-dpctl --dump-dp midonet
1340149 flows
  Flow:
    match keys:
      Tunnel{tun_id=4360, ipv4_src=10.11.0.16, ipv4_dst=10.11.0.15, tun_flag=0,
 ipv4_tos=0, ipv4_ttl=-3}
      InPort{1}
      Ethernet{src=02:13:38:97:08:f3, dst=fa:16:3f:92:53:60}
      EtherType{0x800}
      KeyIPv4{src=8.8.8.8, dst=10.17.3.14, proto=17, tos=0, ttl=55, frag=0}
      UDP{src=53, dst=56975}
    actions:
      Output{port=21}
[...]
```

```
# mm-ctl --list-hosts
Host: id=17ef018f-de8b-431b-89f0-b5472f176769
  name=hostname
  isAlive=true
  addresses:
  vport-host-if-bindings:
    VirtualPortMapping{virtualPortId=ac0c2557-9fa0-4009-9e18-dc62ea65052a,
 localDeviceName='tapac0c2557-9f'}
    VirtualPortMapping{virtualPortId=c37d8bf2-d008-464e-a688-0627f2da342f,
 localDeviceName='f58b0880_MN_dp'}
    VirtualPortMapping{virtualPortId=7aa08012-d06c-4c78-aee8-1fff7c063fed,
 localDeviceName='tap7aa08012-d0'}
    VirtualPortMapping{virtualPortId=5aa6a752-57f2-4749-b160-9e632e0a16bb,
 localDeviceName='f58b0880_MN_dp'}
[...]
```

MTU

アンダーレイネットワークのフラグメンテーションを避けるために、VMインスタンスのMTUは、トンネルプロトコルのオーバーヘッドに対応している必要があります。

この調整されたMTUは、DHCPを通じて、MidoNetに酔って自動的にアドバタイズされます。しかし、これは、VMで使われているオペレーティングシステムによっては、適応されない場合があります。

VMのMTUはアンダーレイのMTUに基づいて設定されていることを確認ください。

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

トポロジーシュミレーション

トポロジーシュミレーションは、ネットワークステートデータベース(NSDB)から仮想トポロジーデータを取り出すMidoNetエージェント(Midolman)によって実行されます。

/var/log/midolman.log ファイルを確認して、エラーか警告がないかを確認してください。

NSDBに対する接続が正常に動いているか確認してください。NSDBは二つのコンポーネント、ZooKeeperとCassandraによって構成されています。

NSDBホストをピングしたり、適切なサービスポートにテルネットすることで、ネットワークのアクセシビリティを手動で検証することができます。

Service	Port
ZooKeeper	2181
Cassandra	9042

仮想トポロジー

仮想トポロジーはNeutronデータベースとMidoNetのネットワークステートデータベース (NSDB)に格納されます。

以下は、確認する一般的なものです。

セキュリティーグループ

理想的なトラフィックパスを確立するために、Neutronセキュリティーグループは設定されていますか？

プロトコル（ICMP, SSH, HTTPなど）と使われているポートに関する適切なルールが存在しているかをチェックしてください。

第2章 共通トピック

目次

MidoNet エージェント

デバッギング

トラブルシューティングの期間に、mn-conf(1) ツールを通じて、ログレベルを増やすことができます。

ログレベルを以下の値に変更できます。 DEBUG, INFO, WARN, ERROR

現在設定されたレベルをチェックするため、下記のコマンドを実行して下さい。

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

レベルを増やすため、下記のコマンドを実行して下さい。

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

この変更を行うために、エージェントの再起動は必要ありません。

MidoNet API

デバッギング

トラブルシューティングの期間に、/usr/share/midonet-api/WEB-INF/classes/logback.xml ファイルを通じて、ログレベルを増やすことができます。

これをするには、ルートのロガーレベルを以下の値に変更してください。 DEBUG, INFO, WARN, ERROR

```
<logger name="org.midonet.event" level="DEBUG">
    <appender-ref ref="EVENT-FILE"/>
</logger>

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

この変更を行うために、Tomcatの再起動が必要です。

ボーダーゲートウェイプロトコル (BGP)

このセクションはBGPに関連する一般的なトピックをいくつかカバーしています。

BGPネームスペースを見つけてください。

```
# ip netns list
mbgpX_ns
```

BGPネームスペースのインターフェースをみる。

```
# ip netns exec <mbgpX_ns> ip link show
```

BGPピアがエンドにある場合でもリーチできることを確認ください。

```
# ip netns exec <mbgpX_ns> ip neigh show
```

BGPピアリングにおけるスニッフ

```
# ip netns exec <mbgpX_ns> tcpdump -i <ns_itf>
```

Quagga’s VTYシェルを入力ください。

```
# ip netns exec <mbgpX_ns> vtysh
```

QuaggaのVTYシェル中の既存の設定を見ます。

```
# 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 192.168.107.29
 network 42.159.202.0/24
 neighbor 192.168.107.30 remote-as 65534
 neighbor 192.168.107.30 timers 5 15
 neighbor 192.168.107.30 timers connect 10
!
line vty
!
end
```

QuaggaのVTYシェル中のBGPサマリーを見ます。

```
# show ip bgp summary
BGP router identifier 192.168.107.29, 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
192.168.107.30  4 65534    5512    5523        0    0    0 07:40:09       10
```

Quagga’s VTYシェル中のBGPルーティング情報を見ます。

```
# show ip bgp
BGP table version is 0, local router ID is 192.168.107.29
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          192.168.107.30           0             0 65534 ?
*> 1.1.1.0/30       192.168.107.30           0             0 65534 ?
```

```
*> 42.159.202.0/24    0.0.0.0           0          32768 i
*> 192.168.0.0        192.168.107.30    0              0 65534 ?
*> 192.168.1.0        192.168.107.30    0              0 65534 ?
*> 192.168.2.0        192.168.107.30    0              0 65534 ?
*> 192.168.49.0       192.168.107.30    0              0 65534 ?
*> 192.168.107.4/30   192.168.107.30    0              0 65534 ?
*> 192.168.107.12/30  192.168.107.30    0              0 65534 ?
*> 192.168.107.20/30  192.168.107.30    0              0 65534 ?
*> 192.168.107.28/30  192.168.107.30    0              0 65534 ?

Total number of prefixes 11
```

ZooKeeper

サーバーがノンエラーステートで走っていることを確認します。もしサーバーが走っている場合は、imokと返してきます。走っていない場合は、何の返りもありあせん。

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

パフォーマンスと接続しているクライアントの統計情報をリスト化します。

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

ZooKeeperデータベースのコンテンツをプリティプリントのテキスト・ファイルにダンプします。

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

ZooKeeperデータベースのコンテンツをマシンが読み取り可能JSONファイルにダンプします。

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

VM の相互接続

状況

VM1 が TCP トラフィックを VM2 に送信できません。 パケットがどこまで到達してから失われたかを特定する必要があります。

物理コンピューティングホストを特定する

どの物理コンピューティングホストに VM が配置されているかを調べるには、Horizon に管理ユーザーとしてログインして、インスタンスページに移動します。

VM をリスト内で検索し、コンピューティングホスト、内部 IP、フローティング IP を書き留めておきます。

VM1: compute1, 192.168.0.1, 172.16.0.1

VM2: compute2, 192.168.0.2, 172.16.0.2

インターフェースを特定する

TAP インターフェース名は、文字列「tap」で始まり、VM のポート UUID の最初の 11 文字が続きます。

VM のポート UUID を特定するには、Horizon で VM のネットワークに移動し、ポートリストで VM の内部 IP を検索します。そこから、次の例のように TAP インターフェース名を作成します。

ポート UUID: 7aa08012-d06c-4c78-aee8-1fff7c063fed

TAP インターフェース: tap7aa08012-d0

インターフェースでトラフィックを検証する

コンピューティングホストの VM の TAP インターフェースで tcpdump を使用すれば、ゲストホストにログインせずに VM の仮想 NIC でトラフィックが確認されているかどうかを検証できます。

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

TAP インターフェースでパケットカウンターを監視します。

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

第3章 ツールとコマンド

目次

このセクションは、便利なツールとコマンドに関する概要を提供します。

midonet-cli

midonet-cli コマンドは、MidoNet APIに接続し、python-midonet-client パッケージをインストールしているホスト上で走らせることができます。

mm-dpctl

mm-dpctl コマンドは、どのMidoNetエージェントでも走らせることができ、現在のフローなどデータパス情報を表示することができます。

利用可能なオプション:

```
$ mm-dpctl
usage: mm-dpctl
    --add-dp <arg>      新しいデータパスを追加
    --delete-dp <arg>   データパスを削除
    --dump-dp <arg>     与えられたデータパスにインストールされている全てのフローを表示
    --list-dps          インストールされているデータパスをリスト化
    --show-dp <arg>     与えられたのデータパスに関連する情報を表示
    --timeout <arg>     何秒でタイムアウトするかを特定。特定した時間内にプログラムが結果が
帰ってこない場合は、停止してエラーコードを戻します。
```

例題:

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

mm-trace

a mm-trace コマンドは、MidolManエージェントがある特定のトラフィックフローをキャプチャして、シュミレーションの各ステージのログを取ることができます。

この設定は、MidolMan再起動の際に、常に同じではありません。

アウトプットは /var/log/midolman/mm-trace.log ファイルに書き込まれます。

利用可能なオプション:

```
$ mm-trace --help
  -h, --host  <arg>   Host (default = localhost)
  -p, --port  <arg>   JMX port (default = 7200)
      --help          ヘルプメッセージを表示

Subcommand: add - トレーシングマッチのパケットを追加
```

```
  -d, --debug               デバグレベルでのログ
      --dst-port  <arg>     TCP/UDP 送信先ポートをマッチ
      --ethertype  <arg>    イーサータイプをマッチ
      --ip-dst  <arg>       送信先ipアドレスをマッチ
      --ip-protocol  <arg>  ipプロトコルフィールドをマッチ
      --ip-src  <arg>       ipソースアドレスをマッチ
  -l, --limit  <arg>        このトレースを無効化する前のマッチするパケット数
      --mac-dst  <arg>      送信先MACアドレスをマッチ
      --mac-src  <arg>      送信元MACアドレスをマッチ
      --src-port  <arg>     TCP/UDPソースポートをマッチ
  -t, --trace               トレースレベルでのログ
      --help                ヘルプメッセージを表示
Subcommand: remove - トレーシングマッチのパケットを除去
  -d, --debug               デバグレベルのログ
      --dst-port  <arg>     TCP/UDP 送信先ポートをマッチ
      --ethertype  <arg>    イーサータイプをマッチ
      --ip-dst  <arg>       送信先ipアドレスをマッチ
      --ip-protocol  <arg>  ipプロトコルフィールドをマッチ
      --ip-src  <arg>       ipソースアドレスをマッチ
  -l, --limit  <arg>        このトレースを無効化する前のマッチするパケットの数
      --mac-dst  <arg>      送信先MACアドレスをマッチ
      --mac-src  <arg>      送信元MACアドレスをマッチ
      --src-port  <arg>     TCP/UDPソースポートをマッチ
  -t, --trace               トレースレベルでのログ
      --help                ヘルプメッセージを表示
Subcommand: flush - トレーシングがマッチしているリストをクリア
  -D, --dead-only   期限切れのトレーサーのみをフラッシュ
      --help        ヘルプメッセージを表示
Subcommand: list -  アクティブなトレーシングマッチをリスト化
  -L, --live-only   アクティブなトレーサーのみをリスト化
      --help        ヘルプメッセージを表示
```

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

ip コマンドは、ルーティング、デバイス、ポリシールーティング、トンネルを表示したり、操作したりするために使用します。

詳細情報に関するメインページはこちらです： http://linux.die.net/man/8/ip

インターフェースをリスト化します。

```
# ip link show
```

名前空間をリスト化します。

```
# ip netns list
```

名前空間の中で、インターフェースをリスト化します。

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

第4章 ディレクトリーとファイル

目次

このセクションは、よく使う設定とログファイルの概要を提供します。

ファイル名とパスは、使っているオペレーティングシステムとOpenStackのディストリビューションによって、微妙に異なりますので注意してください。

Cassandra

File	Type
/etc/cassandra/conf/cassandra.yaml	CONF
/var/log/cassandra/cassandra.log	LOG

MidoNet Agent

File	Type
/etc/midolman/midolman-akka.conf	CONF
/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 API

File	Type
/usr/share/midonet-api/WEB-INF/web.xml	CONF
/var/log/tomcat/catalina.out	LOG
/var/log/tomcat/midonet-api.log	LOG

Quagga (BGPD)

File	Type
/var/log/quagga/bgpd.log	LOG

ZooKeeper

File	Type
/etc/zookeeper/zoo.cfg	CONF

File	Type
/var/log/zookeeper/zookeeper.out	LOG

第5章 プロセス

このセクションは、共通プロセスを提供します。

使っているオペレーション・システテとOpenStackディストリビューションによって、名前とファイルパスが異なります。

プログラ	プロセス
Cassandra	java […] org.apache.cassandra.service.CassandraDaemon
MidoNet Agent	java […] org.midonet.midolman.Midolman
MidoNet Agent (Watchdog)	/usr/bin/python /usr/bin/wdog […] org.midonet.midolman.Midolman
MidoNet API (Tomcat)	java […] org.apache.catalina.startup.Bootstrap
ZooKeeper	java […] org.apache.zookeeper.server.quorum.QuorumPeerMain