

MidoNet 運用 ガイド

2015.06-SNAPSHOT (2015-11-18 07:27 UTC)

DRAFT



midonet

docs.midonet.org

T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

iv

図の一覧

15.1. Topology with VLANs and L2 Gateway 106

表の一覧

2.1.	Keystone Service Protocols	8
2.2.	Admin Roles	9
7.1.	CLIルールチェーン属性	36
7.2.	CLIルールチェーン属性のうちパケットとマッチするもの	36
13.1.	Configuration Files/Locations	78
13.2.	Event Message Files/Locations	78
17.1.	推奨される設定値	113

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。

目次

このセクションでは、MidoNetが利用できるクラウドから、外部ネットワークに向けるアップリンクの設定のしかたを記しています。

1. 仮想ポートと外部ポートを接続します。
2. ルーターを設定します。これには、ネットワーク間のトラフィックルートの設定も含まれます。
3. ダイナミックルーティングを設定することで、ローカルの自律システム(AS)と他の自律システムの間のルート交換を行えるようになります。MidoNetはBGPをサポートしており、これはフローティングIPなどと関連づけるネットワークへのルートなど、MidoNetをアドバタイズをすることを許可する外部のルーティングプロトコルです。また、BGPのピアからの到達可能性情報やルートを受け取ります。

BGP 設定

通常、二つの独立したアップリンクルーターを通じて、MidoNetネットワークをインターネットに接続します。シンプルなケースでは、MidoNetを二つのBGP利用可能なルーターを通じてインターネットに接続します。これは仮想ルーターで二つのポートを作成して、二つの違ったホスト（二つのゲートウェイノード）でネットワークインターフェースにバインドする最適な方法です。これによって、ゲートウェイノードホスト間でロードを分散でき、単一障害点を削除することができます。二つのポートは、ステートフルポートペアとして設定される必要があります。詳細に関しては、「[ステートフルポートグループ](#)」[\[19\]](#)を参照ください。



一つのBGPセッションは仮想ルーターポートに関連づけられています。MidoNetは現在、各ポートに一つのBGPセッションを行う設定のみをサポートしています。

MidoNetは仮想ルーターのために、BGPセッションを終了する為にquaggaの bgpd を使用しています。Bgpdがそのピアから学ぶルートはMidoNetトポロジーの仮想ルーターに追加されます。 QuaggaパッケージはMidoNetリリースパッケージレポジトリで提供されます。Midolmanを走らせているシステムは、BGPを設定する全てを保持している必要があります。ある特定の仮想ルーターが向かっているホストでbgpdプロセスが走っている必要があります。



BGPを設定する前に、BGPセッションのためのローカルとピアの自律システム (AS) 番号とBGPピアのIPアドレスを確認してください。

DIT

4


```
host host0 interface veth1 port router0:port
```

7. 外部インターフェースにマスカレードを加えます。 ”フェイクの” 外部ネットワークに属するアドレスのオーバーレイから来る接続がNAT化されます。パケットが転送できることを確認してください。

```
# iptables -t nat -I POSTROUTING -o eth0 -s 200.200.200.0/24 -j MASQUERADE
# iptables -I FORWARD -s 200.200.200.0/24 -j ACCEPT
```

フローティングIPを使って、アンダーレイホストからVMへのリーチが可能になりました。VMも外部リンクにリーチできるようになります。（ホストが外部接続性をもっている場合に限ります）

第2章 認証及び承認

目次

MidoNet内で使用可能な認証サービス	7
MidoNet内のロール	8
Keystone認証サービスの使用	9

MidoNetアプリケーションプログラミングインターフェース (API) は外部の識別サービスと一体化して認証及び承認サービスを提供します。

このAPIはテナントの作成や削除はしませんが、リソースの識別や、テナントIDを使用してクエリにフィルターをかけます。テナントは外部の識別サービスにより完全管理され、必要に応じて、テナントIDの文字列表現はMidoNet APIに送付されます。このデザインにより、MidoNet APIを含むクラウド環境内で、ひとつの識別サービスが認証及び承認を全てのサービスに提供する、連合された識別サービスモデルが可能になります。

MidoNet APIは独自の識別サービスはもっていませんが、シンプルな認証(ユーザー確認のため)と承認(ユーザーのアクセスレベルをチェックするため)機能をもっています。認証に関しては、HTTPヘッダーに含まれたトークンを外部の識別サービスに転送します。新しいトークンを作る場合には、認証情報であるユーザーネームとパスワードを使用して、APIに外部の識別サービスにログインさせます。(トークンについての詳しい情報は、MidoNet REST APIドキュメントを参照してください) 承認に関しては、MidoNet APIは次のセクションにて説明があるとおり、シンプルなロールベースアクセスコントロール(RBAC)メカニズムを提供しています。

MidoNetのAPIを使った認証及び承認の実装に関する情報については、MidoNet REST APIドキュメントをご参照ください。OpenStack APIに関する情報につきましては、<http://docs.openstack.org/>よりDocumentation and APIのリンクを参照してください。

MidoNet内で使用可能な認証サービス

MidoNet APIには、KeystoneAuthServiceとMockAuthServiceの2種類の認証モードがあります。

このセクションでは、Keystoneの認証サービスと模擬認証、そしてweb.xmlファイルを使いどのようにして必要なサービスを選択するのかについて説明します。

Keystone特有の設定

OpenStack Keystone認証サービスをMidoNetで使用するために、web.xmlファイルを設定する必要があります。

認証サービスのためにKeystoneを規定についての説明です。設定要素の名前: keystone-service_protocolとなり、認められた値: http, httpsとなります。プレーンテキストのHTTPを使い、httpを使ってKeystoneにアクセスすることが可能になります。httpsを規定した場合、MidoNet APIサーバーとKeystone認証サーバーの接続は暗号化され、httpsが推奨されます。下記の例は、Keystoneを使って暗号化されたコ

コミュニケーションの設定に使われた、XML内でエンコードされた名前と値キーのペアです。

表2.1 Keystone Service Protocols

Parameter Name	Value	Description
	keystone-service_protocol	keystone-service_protocol
http	Keystoneサーバーと通信するため通常のHTTPを使用	

Parameter Name	Value	Description
keystone-service_protocol	http	Keystoneサーバーと通信するため通常のHTTPを使用
	https	Keystoneサーバーと通信するため暗号化されたHTTPSを使用

必要なサービスプロトコルを含むため、/usr/share/midonet-api/WEB-INF/web.xml ファイルを編集してください。

```
<context-param>
  <param-name>keystone-service_protocol</param-name>
  <param-value>https</param-value>
</context-param>
```

模擬認証について

模擬認証は、`web.xml` の設定ファイル内にある全てのロールにトークンをマッピングすることにより、認証システムをまねるものです。もし、アドミンロールにマッピングされたトークンがAPIリクエストに使われると、認証と承認は無効にされます。



警告

このモードはテスト目的で使用するもので、プロダクションでは使用できません。

MidoNet内のロール

MidoNet APIは承認を行うためRBACメカニズムを実装しています。

AutoRoleクラスにて定義されるMidoNet内のロールは以下のとおりです。

- ・ アドミン: システムのルートアドミニストレーターです。このロールを持ったユーザーは全ての運用を行うことが許されています。
- ・ テナントアドミン: このロールを持ったユーザーは自分のリソースに対して読み書きのアクセス権を持っています。
- ・ テナントユーザー: このロールを持ったユーザーは自分のリソースに対してリードアクセスのみを持っています。



注記

外部の識別サービスにて定義されたRBACポリシーは、MidoNet RBAC内では適用されないことに留意してください。たとえば、Keystone内で持っているアクセスタイプが、そのままMidoNet内で同じアクセスを持つわけではありません。MidoNet APIは上記に記載されている3つのロールへのポリシーに準じています。

MidoNet用アドミンロールの作成

承認サービスはロールマッピングを決めるにあたり、web.xmlファイルに明記された入力内容に依存しています。web.xmlはMidoNet APIの設定ファイルであり、以下のロケーションにあります。

```
/usr/share/midonet-api/WEB-INF/web.xml
```

web.xml file内で設定する設定要素は、名前と値のペアにより構成されています。名前と値のペアを加える場合には、XMLにてエンコードしてください。

外部のサービス(OpenStack Keystoneのような)内でのロールを、MidoNetでのロールに変換するために承認サービスを使うことができます。下記の例は、別々のアドミンロール(auth-admin-role、auth-tenant-admin、auth-tenant_user_role)をどのようにしてMidoNet向けに作成するかを表しています。

表2.2 Admin Roles

Name	Value	Description
auth-admin_role	[name]	Specifies the name for the admin role in MidoNet.
auth-tenant_admin_role	[name]	Specifies the name for the tenant admin role in MidoNet.
auth-tenant_user_role	[name]	Specifies the name for the tenant user role in MidoNet.

```
...
<context-param>
  <param-name>auth-admin_role</param-name>
  <param-value>mido_admin</param-value> </context-param>
<context-param>
  <param-name>auth-tenant_admin_role</param-name>
  <param-value>mido_tenant_admin</param-value> </context-param>
<context-param>
  <param-name>auth-tenant_user_role</param-name>
  <param-value>mido_tenant_user</param-value>
</context-param>
...
```

上記の例において、外部サービスに保管されているmido_admin、mido_tenant_admin及びmido_tenant_userロールはそれぞれMidonet内admin、tenant_admin及びtenant_user in Midonetと変換されます。

Keystone認証サービスの使用

このセクションでは、MidoNetでのKeystone認証サービスの使用方法を説明します。

Keystone認証の有効化

MidoNetでOpenStack Keystone認証サービスを使うためには、web.xmlファイル内にいくつかの設定をする必要があります。

auth-auth_provider

認証サービスを提供するJavaクラスの、完全修飾パスをリスト化します。

```
<context-param>
  <param-name>auth-auth_provider</param-name>
```

12

13

```
midonet> list host
host host0 name controller alive true
host host2 name compute1 alive true
host host3 name compute3 alive false
host host1 name compute2 alive false
```

- 下記のソース例にあるように、特定のホスト上のインターフェイスをリストアップするコマンドを入力します。

```
midonet> host host0 list interface
iface midonet host_id host0 status 0 addresses [] mac 12:6e:b7:d0:4f:f1 mtu 1500 type
Virtual endpoint DATAPATH
iface lo host_id host0 status 3 addresses [u'127.0.0.1', u'0:0:0:0:0:0:0:1'] mac
00:00:00:00:00:00 mtu 65536 type Virtual endpoint LOCALHOST
iface tapbf954474-ef host_id host0 status 3 addresses
[u'fe80:0:0:0:dc40:9aff:feef:7b5e'] mac de:40:9a:ef:7b:5e mtu 1500 type Virtual
endpoint DATAPATH
iface eth0 host_id host0 status 3 addresses [u'192.168.0.3',
u'fe80:0:0:0:f816:3eff:febe:590'] mac fa:16:3e:be:05:90 mtu 8842 type Physical endpoint
PHYSICAL
```

- ・ 下記のソース例にあるように、特定のホストにポートを閲覧するコマンドを入力します。

```
midonet> host host0 list binding
host host0 interface tapbf954474-ef port bridge0:port0
```

上記のアウトプットされたソースは、host0上のデバイス tapbf954474-ef は、bridge0上のport0に現在接続されていることを示しています。

ホストの認証

新しいホストをトンネルゾーンへ追加します。トンネルゾーンへホストを認証する場合、下記の方法で行います。

1. 全てのトンネルゾーンを閲覧するには、`list tunnel-zone`というコマンドを入力します。例としては下記のようなソースが挙げられます。

```
midonet> list tunnel-zone
tzone tzone0 name gre type gre
```

2. 全てのホストを閲覧するには、`list host`というコマンドを入力します。例としては下記のようなソースが挙げられます。

```
midonet> list host
host host0 name compute-1 alive true
host host1 name compute-2 alive true
```

3. ホスト上の全てのインターフェイスをリストアップするには、``host hostX list interface``というコマンドを入力します。（コマンド内のXは適切なホストエイリアスへダイナミックにアサインされる数字を示しています。）

```
midonet> host host0 list interface
iface lo host_id host0 status 3 addresses [u'127.0.0.1', u'0:0:0:0:0:0:0:1'] mac
00:00:00:00:00:00 mtu 65536 type Virtual endpoint LOCALHOST
iface midonet host_id host0 status 0 addresses [] mac 8e:4d:60:c1:70:d7 mtu 1500 type
Virtual endpoint DATAPATH
iface eth1 host_id host0 status 3 addresses [u'fe80:0:0:0:250:56ff:fe93:7c35'] mac
00:50:56:93:7c:35 mtu 1500 type Physical endpoint PHYSICAL
iface eth0 host_id host0 status 3 addresses [u'10.1.2.200',
u'fe80:0:0:0:250:56ff:fe93:c9a4'] mac 00:50:56:93:c9:a4 mtu 1500 type Physical
endpoint PHYSICAL
```


ホストの削除

アクティブでないホストを削除するには、この方法で行います。

1. ホストをリストアップするコマンドを入力します。

```
midonet> list host  
host host0 name precise64 alive true
```

2. エイリアスに特定されたホストを削除するコマンドを入力します。

```
midonet> host host0 delete
```

17


```
midonet> router router1 add port address 10.100.1.1 net 10.0.0.0/24
router1:port0
```

+ 上記のアウトプットは、新しいポートにエイリアス（"port0"）をアサインしていることを示しています。

1. ルーターへのポート情報をリスト化する為にコマンドを入力します。

```
midonet> router router1 list port
port port0 device router1 state up mac 02:a6:81:08:ab:5d address 10.100.1.1 net 10.0.0.0/24
```

上記のアウトプットは以下を示しています。

- ・ ポート(“port0”)にアサインされたエイリアス
- ・ (router1)にアタッチされたデバイス、ポート
- ・ ポートの状態 (up)
- ・ ポートのMACアドレス
- ・ ポートのIPとネットワークアドレス

ブリッジの追加

このプロシージャを使ってブリッジを作成します。

ブリッジを作成して名前をアサインするために、以下のコマンドを入力します。

```
midonet> bridge create name test-bridge
bridge1
```

上のアウトプットは、エイリアス（" bridge1" ）が新しいブリッジにアサインされたことを示しています。

ブリッジにポートを追加

ブリッジにポートを加える為に、このプロシーダを使います。

1. 現在のテナントのブリッジをリスト化するためにコマンドを入力します。

```
midonet> bridge list
bridge bridge1 name test-bridge state up
```

2. 適切なブリッジにポートを加える為のコマンドを入力します。

```
midonet> bridge bridge1 add port  
bridge1:port0
```

上のアウトプットはエイリアス("port0")が新しいポートにアサインされたことを示しています。

外部ポートをホストにバインディング

MidoNetが利用可能になったクラウドを外部ネットワークに接続する為に、ホストに外部ポートをバインドする必要があります。例えば、ネットワークインターフェースカード（NIC）とeth0のIDなどです。

1. ホストをリスト化するためのコマンドを入力します。

```
midonet> list host
host host0 name compute-1 alive true
host host1 name compute-2 alive true
```

2. 現在のテナントのブリッジをリスト化するためのコマンドを入力します。

```
midonet> list bridge
bridge bridge0 name External state up
bridge bridge1 name Management state up
bridge bridge2 name Internal state up
```

3. 適切なブリッジにポートをリスト化するためのコマンドを入力します。

```
midonet> bridge bridge0 list port
port port0 device bridge0 state up
port port1 device bridge0 state up
port port2 device bridge0 state up
```

4. ある特定のホスト向けのインターフェースをリスト化するコマンドを入力します。

```
midonet> host host0 list interface
iface lo host_id host0 status 3 addresses [u'127.0.0.1', u'0:0:0:0:0:0:0:1'] mac
00:00:00:00:00:00 mtu 65536 type Virtual endpoint LOCALHOST
iface midonet host_id host0 status 0 addresses [] mac 8e:4d:60:c1:70:d7 mtu 1500 type
Virtual endpoint DATAPATH
iface eth1 host_id host0 status 3 addresses [u'fe80:0:0:0:250:56ff:fe93:7c35'] mac
00:50:56:93:7c:35 mtu 1500 type Physical endpoint PHYSICAL
iface eth0 host_id host0 status 3 addresses [u'10.1.2.200',
u'fe80:0:0:0:250:56ff:fe93:c9a4'] mac 00:50:56:93:c9:a4 mtu 1500 type Physical
endpoint PHYSICAL
```

5. あるホストを仮想ポートにバインドする為のコマンドを入力します。

```
midonet> host host0 add binding
host interface port
```

6. ホストの物理インターフェースとブリッジの仮想ポートをバインドするためのコマンドを入力します。

```
midonet> host host0 add binding port bridge0:port0 interface eth1
host host0 interface eth1 port bridge0:port0
```

ステートフルポートグループ

MidoNetはステートフルなポートグループを特徴としています。これは、通常ロードバランスやリンク冗長の実行を行うために、論理的に関連づけられた仮想ポートのグループ（通常は2つ）です。

そのようなポートに対して、MidoNetは接続の二つのエンドポイントの状態をローカルとしてキープします。ほとんどの場合、MidoNetを横切る接続はポートのシングルペアの間で、その状態をキープします。二つのアップリンクBGPポートとルーター、もしくは、物理L2ネットワークを二つポートがあるL2GWを結びつけるような典型的なケースがあります。これらのケースでは、ポートのペアがポートのセットになりますが、それはパケットが違ったパスを通じてリターンされるためです。これらのポートペアは状態を共有します。

ポートグループコマンドを使って、MidoNet CLIでステートフルなポートグループを設定します。

T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

DIT

上記のアウトプットはrouter1のport0が bridge1のport0に接続されたことを示しています。

二つのルーターの接続

各ルーターの仮想ポートを通じて、二つの仮想ルーターを簡単に繋げることができます。

二つのルーターにルーターポートを作成して、同じサブネットにポートを割り当てることを確認してください。ルーターの作成とルータポートの追加に関する情報は[4章 デバイスの抽象化 \[17\]](#)を参照ください。

二つのルーターを繋げるには、

1. 現在のテナントのルーターをリスト化するためのコマンドを入力してください。 +

```
midonet> list router
router router3 name test-router2 state up
router router1 name test-router state up
```

2. 接続したいルーターの一つのポートをリスト化するコマンドを入力してください。

```
midonet> router router1 list port
port port0 device router1 state up mac 02:a6:81:08:ab:5d address 10.100.1.1 net 10.0.0.0/24 peer bridgel:port0
port port1 device router1 state up mac 02:fa:5f:87:bb:d2 address 10.100.1.2 net 10.0.0.0/24
```

3. 接続したいルーターの一つのポートをリスト化するコマンドを入力してください。

```
midonet> router router3 list port
port port0 device router3 state up mac 02:df:24:5b:19:9b address 10.100.1.128 net 10.0.0.0/24
```

4. あるルータのポート（例えば、router1のport1）から、別のルータのポート（例えば、router3のport0）にバインドする為のコマンドを入力してください。

```
midonet> router router1 port port1 set peer router3:port0
```

5. ルーターの一つのポートをリスト化するコマンドを入力してください。 +

```
midonet> router router1 list port
port port0 device router1 state up mac 02:a6:81:08:ab:5d address 10.100.1.1 net 10.0.0.0/24 peer bridge1:port0
port port1 device router1 state up mac 02:fa:5f:87:bb:d2 address 10.100.1.2 net 10.0.0.0/24 peer router3:port0
```

上記のアウトプットは、router1のport1とrouter3のport0が繋がったことを示しています。

- ## タイプ

送信先

ネクストホップゲートウェイ

25

通常のルーター（＝物理的なルーター）のルートはターゲット/デスティネーション仮想ポートを持っていない（MidoNetの' Normal' ルートは持っています。”タイプ”をご参照ください）という点において、MidoNetの仮想ルーターとは違うということを留意してください。従って、通常のルーターはパケットを放つのにどのポートが使われるべきかを決めるために、ネクストホップゲートウェイIPアドレスを使います（ポートのプレフィックスはネクストホップゲートウェイのIPアドレスにマッチします）。

ネクストホップポート

ピアデバイスに接続されているポートのIDを表示します。

ウェイト

複数のパスがあるデスティネーションのロードバランシングに使われます。高いウェイト値は望ましいパス（例えば、高い帯域幅）を識別します。デフォルトのウェイト値は100です。”ソース”もご参照ください。

ルートの表示

MidoNet内のそれぞれの仮想ルーターで定義されたルートを表示できます。例えば、仮想ブリッジや、テナントルーターやMidoNet Providerルーターのような他のルーターへのルートについてのインフォメーションを表示できます。

現在のテナントのルーターについてのインフォメーションを表示するため、

1. 現在のテナントへのルーターをリスト化するため下記のコマンドを入力します。

```
midonet> list router
router router0 name tenant-router state up infiltr chain0 outfilter chain1
```

2. ルーターへのルートリスト、この場合はテナントルーター（ルーター0）をリスト化するため下記のコマンドを入力します。

```
midonet> router router0 list route
route route0 type normal src 0.0.0.0/0 dst 169.254.255.2 port router0:port0 weight 0
route route1 type normal src 0.0.0.0/0 dst 0.0.0.0/0 port router0:port0 weight 100
route route2 type normal src 0.0.0.0/0 dst 172.16.3.0/24 port router0:port1 weight 100
route route3 type normal src 172.16.3.0/24 dst 169.254.169.254 gw 172.16.3.2 port
  router0:port1 weight 100
route route4 type normal src 0.0.0.0/0 dst 172.16.3.1 port router0:port1 weight 0
```

ルートリストは下記のインフォメーションを示します。

- トラフィックをマッチするためのソース(src)。ルート3はマッチする特定のソースネットワークを示します。 0.0.0.0/0は全てのネットワークからのトラフィックとマッチするという意味です。
- このトラフィックのデスティネーション(dst)。これはネットワークまたは特定のインターフェースとなります。
- ルート0は特定のインターフェースへのルートの例を表示します。これは、link-localアドレスへのルートで、この例で言うと、MidoNet プロバイダールーターで見られます。
- ルート2は172.16.3.0/24ネットワークへのルートを示し、そしてこのネットワークはプライベートネットワークとなります。

-
- 27

```
midonet> show bridge bridge0
bridge bridge0 name demo-ext-net state up
```

demo-ext-netは外部のパブリックネットワークです。

ブリッジ0に関するインフォメーションをリスト化するにはコマンドを入力します。

```
midonet> bridge bridge0 list port
port port1 device bridge0 state up
port port2 device bridge0 state up
port port0 device bridge0 state up peer router1:port1
```

上記のアウトプットが示すのは:

- ブリッジ0には3つのポートがあります。
- ブリッジ0上のポート0はルーター1(MidoNet プロバイダルーター)上のポート1と見られています。

MidoNet プロバイダルータールートを一覧化するには、以下のコマンドを入力します。

```
midonet> router router1 list route
route route0 type blackhole src 0.0.0.0/0 dst 198.51.100.0/24 weight 100
route route1 type normal src 0.0.0.0/0 dst 198.51.100.3 port router1:port0 weight 100
route route2 type normal src 0.0.0.0/0 dst 169.254.255.1 port router1:port0 weight 0
route route3 type normal src 0.0.0.0/0 dst 198.51.100.2 port router1:port0 weight 100
route route4 type normal src 0.0.0.0/0 dst 198.51.100.1 port router1:port1 weight 0
```

下記はこれらのルートに関するいくつかの注意点です。

- ・フローティングIPアドレスが外部ネットワークに存在しない場合、タイプ =ブラックホールであり、トラフィックはブラックホールにルートされ、トラフィックはそこでドロップされます。この場合、マッチする送信元は任意のネットワーク(0.0.0.0/0)で、送信先は外部のパブリックネットワークである198.51.100.0/24ネットワークです。
- ・ルート1はVMのフローティングIPアドレス(198.51.100.3)へのルートを示します。
- ・ルート2はテナントルーターへのlink-localコネクション(169.254.255.1)へのルートを示します。
- ・ルート3は外部のネットワークへのゲートウェイを示します。
- ・ルート4が言っていることは以下になります: 任意の送信元ネットワーク(0.0.0.0/0)と送信先198.51.100.1にマッチするトラフィック (MidoNet Provider Routerの外部ネットワーク(198.51.100/24) (demo-ext-net, bridge0)へのインターフェース) について、このトラフィックをネットワークへのルーターのインターフェース (ブリッジ1) であるポート1に転送してください。

実在のデプロイメントでは、MidoNet プロバイダールーターに接続されたほとんどのポートはアップリンクポートで、通常複数のアンリンクポートがあります(上記で説明されているネットワークの例では、ネットワークはアップリンクポートに接続されていません。) MidoNetの外に接続を持つ他の仮想ポートのように、それぞれのポートはデバイス(イーサネットインターフェース)とホストに接続されています。このホストはサービスプロバイダーへつながるアップストリームルータへと物理的につながっているゲートウェイノードです。

ルートの追加

下記は、ルートの追加をする場合のいくつかの例です。

規定するアトリビュートは、

- dst = マッチする送付先IPアドレスもしくはネットワーク
- src = マッチする送付元IPアドレスもしくはネットワーク
- type = 例としては、"normal"
- port = トラフィックを排出するポート

MidoNet CLIを使ってルートを追加するには、

1. ルートを加えるためにコマンドを入力してください。

```
midonet> router router2 add route dst 169.254.255.0/30 src 0.0.0.0/0 type normal port
router2:port2
router2:router2
```

上記のコマンドは下記のインストラクションを含んでいます。

- 全ての送付元ネットワーク(0.0.0.0/0)を持ち、かつ送付先ネットワーク169.254.255.0/30であるトラフィックについて、このトラフィックをルート2上のポート2に転送してください。



注記

上記のルートを追加する前に、下記のコマンドを使ってルーター2が追加されました。

```
midonet> router router2 add port address 169.254.255.3 net 169.254.255.0/30
router2:port2
```

2. ルートの追加を確定するため、ルーター2にルートをリスト化するコマンドを入力してください。

```
midonet> router router2 list route
route route0 type normal src 0.0.0.0/0 dst 10.100.1.1 port router2:port0 weight 0
route route1 type normal src 0.0.0.0/0 dst 10.100.1.2 port router2:port1 weight 0
route route2 type normal src 0.0.0.0/0 dst 169.254.255.3 port router2:port2 weight 0
route route3 type normal src 0.0.0.0/0 dst 169.254.255.0/30 port router2:port2 weight 0
```

ルートの削除

スタンドアロンSDNコントローラーとしてMidoNetを使っていて、ルートを削除したい場合があります。そのような場合は、物理的なネットワークデバイスの管理に関係します。

例えば、マニュアルでルートを足す必要がある何らかの処理を取り消す場合、ルートを削除することができます。



警告

OpenStack Neutronオペレーションの結果として自動的に加えられたルートを削除することは推奨されていません。

ルートを消すために、

1. 特定のルーターにルートをリスト化するためのコマンドを入力してください。

```
midonet> router router2 list route
route route0 type normal src 0.0.0.0/0 dst 10.100.1.1 port router2:port0 weight 0
route route1 type normal src 0.0.0.0/0 dst 10.100.1.2 port router2:port1 weight 0
route route2 type normal src 0.0.0.0/0 dst 169.254.255.3 port router2:port2 weight 0
route route3 type normal src 0.0.0.0/0 dst 169.254.255.0/30 port router2:port2 weight 0
```

上記は、ルーター2にルートをリスト化するコマンドです。

2. 希望のルーターから希望のルートを削除するコマンドを入力してください。

```
midonet> router router2 delete route route2
midonet> router router2 delete route route3
```

上記のコマンドはルート2とルート3をルーター2より削除します。

3. 削除を確定するためルーター上のルートをリスト化するコマンドを入力してください。

```
midonet> router router2 list route
route route0 type normal src 0.0.0.0/0 dst 10.100.1.1 port router2:port0 weight 0
route route1 type normal src 0.0.0.0/0 dst 10.100.1.2 port router2:port1 weight 0
```

目次

- これらのチェーンはルーター上にのみならず、ブリッジ上、あるいは各種ポート上にも置くことができます。
- プレルーティングチェーンは入口ポートIDへのアクセス権を保有しています。ポストルーティングチェーンは入口ポートと出口ポートの両方にアクセスする権限を持っています。ポストルーティング中は、ルーターがパケットのルート指定を行っており、入口ポートと出口ルーターポートの両方を認知しています。

****そのルールチェーンは、パケットのヘッダーの修正を行なった可能性があります。たとえば、ポートをマスカレードするために行なうことがあります。ポートマスカレード、あるいはNATのためにパケットのヘッダーを変更することは、フローに存在するどのパケットにも適用しています。**

- DROP: ルールチェーンはパケットの処理を停止して、自らの呼び出し元に戻らなければなりません(DROP, new_packet)。
- RETURN: ルールチェーンはパケットの処理を停止し、自らの呼び出し元に戻らなければなりません(CONTINUE, new_packet)。この場合、このチェーンの中ではこれ以上実行されるルールがないため、ACCEPTアクションともCONTINUEのアクションとも流れが異なることに注意をしますが、今呼びかけを行なっているほうのチェーンの中にあるルールが実行されることはあります。
- REJECT: ルールチェーンはパケットの処理を停止して、自らの呼び出し元に戻らなければなりません(REJECT, new packet)。

ルール種別

ACCEPT, DROP, REJECT, RETURN

DNAT, SNAT

このnext_actionは、チェーンをより柔軟に構築することを可能にしてくれます。パケットをマッチさせた後、つまりパケットを修正後(そのアドレスの一部を変換した後)、次にすべきことを選択することは複雑な作業であり、自分次第ということになります。利用できる選択肢は次のとおりです。

-
- 35

[NOTE]

ルールの中で特定したポートは、仮想ルーター上か仮想ブリッジ上の仮想ポートです。仮想ポートは、物理的なホスト上にある特定のイーサネットインターフェース(たとえばtap)に結びついているのかもしれませんが、別の仮想ポートと対等の関係にあるのかもしれません。(その場合には仮想ポートは2つの仮想機器に接続します。) いずれにしても、仮想ポートは仮想のものであると考えるべきです。なぜならば、各種ルールは仮想トポロジにしか存在せず、さらに、ルールを評価している間は、仮想ポートが物理的にイーサネットのインターフェースに結びついているかどうかは全く認知されないからです。

属性	説明
pos <INTEGER>:	チェーンの中におけるルールの位置
type <TYPE>:	The rule <TYPE>; これはほとんどの場合、通常のフィルタリングルールと様々な種類のNATルールとを区別するために使われます。認知された<TYPE>バリューは次のとおりです。accept, continue, drop, jump, reject, return, dnat, snat, rev_dnat, rev_snat.
action accept	continue
return:	このルールアクションはNATルールにとってのみ意味を持ちます。
jump-to <CHAIN>:	(これがもしもジャンプルールである場合)ジャンプして向かっていく先のチェーン
target <IP_ADDRESS[-IP_ADDRESS][:INTEGER[-INTEGER]]>:	dnatルールがあるいはsnatルールである場合にはNAT標的です。少なくともIPアドレス1つは提供しなければなりません。また、このNAT標的は任意で、2つめのアドレスを含めることにより、アドレス範囲とL4ポート番号を形成する、あるいはポートの範囲を形成することもできます。

Attributes That Match Packets	解説
hw-src [!]<MAC_ADDRESS>:	ソースのハードウェアのアドレス
hw-dst [!]<MAC_ADDRESS>:	行き先のハードウェアアドレス
ethertype [!]<STRING>:	このルールによりマッチさせたパケットのデータリンク層(EtherType)を設定します。
in-ports [!]<PORT[, PORT...]>:	現在パケットを処理している仮想機器にパケットが進入する時に通過をした仮想ポートをマッチさせます。
out-ports [!]<PORT[, PORT...]>:	現在パケットを処理している仮想機器からパケットが出ていく時に通過をするポートをマッチさせます。
tos [!]<INTEGER>:	マッチさせるべきパケットのサービス種別フィールド(TOSフィールド)のバリュー。このフィールドは、差別化されているサービスバリューをマッチさせる時に使用してください。詳細につきましては https://www.ietf.org/rfc/rfc2474.txt [TOS]を参照してください。
proto [!]<INTEGER>:	これはマッチさせるべきIPプロトコル番号です。詳しくは次のリンクを参照してください。 Protocol Numbers 事例は次のとおりです: ICMP = 1, IGMP = 2, TCP = 6, UDP = 17
src [!]<CIDR>:	ソースのIPアドレスあるいはCIDRブロック
dst [!]<CIDR>:	行き先のIPアドレスあるいはCIDRブロック
src-port [!]<INTEGER[-INTEGER]>:	TCPソースポートあるいはUDPソースポートあるいはポートの範囲
dst-port [!]<INTEGER[-INTEGER]>:	TCPポートあるいはUDP行き先ポートあるいはポートの範囲

T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

=条件例 1

その1つの方法が、「条件」を持ったDROPルールあるいはREJECTルールを構築する方法です。また、これらの属性を保有したACCEPTルールは下に挙げた意味を持ちます。

- 

ルール2が意味を持つには無条件dropが必要です。

必要であればsetコマンドを使うかあるいは他の手段を使って、適切なテナントにアクセスします。 +

コマンドを入力して新しいルールチェーンを作成し、そのルールチェーンに名前を付与します。 +

1. ソース10.0.0.0/16を持たないIPv4トラフィックをドロップするコマンドを入力します。

行き先が10.0.5.0/24を持ったIPv4トラフィックを受け入れるコマンドを入力します。

1. 新しいルールチェーンに追加されたルールをリスト化するためのコマンドを入力します。

```
midonet> chain chain5 list rule
rule rule3 ethertype 2048 src !10.0.0.0/16 proto 0 tos 0 pos 1 type drop
rule rule2 ethertype 2048 dst 10.0.5.0/24 proto 0 tos 0 pos 2 type accept
```

条件例 2

条件の事例1と同様ですが、条件の事例2の場合はルール構築の仕方が異なることを前提にしています。全てのパケットとマッチするようなDROPルールを1つ、チェーンの終わりに保有するためです。チェーンの初めの箇所には、通過を許可されたパケット/フローとマッチするACCEPTルールを設置します。

条件の事例 1 が許可したトラフィック用のACCEPTルールは、以下に挙げる属性を持った「条件」を持っています。

ルール言語の中では、チェーンは以下を保有します。

ACCEPT when src=(10.0.0.0, 16) OR dst=(10.0.5.0, 24)

ルールの終わりの箇所

その他のパケットは全てDROPします。

上記した属性を持ったルールチェーンを作成する時には次の通りにします。

必要があればアイには、`sett`コマンドを用いるか、あるいはその他の手段を使って、適切なテナントにアクセスします。・ +

```
midonet> sett 10a83af63f9342118433c3a43a329528
tenant id: 10a83af63f9342118433c3a43a329528
```

新しいルールチェーンを作成するためにコマンドを入力し、そのルールチェーンに名前を付与します。・+

```
midonet> chain create name "accept_src_dst_mynetwork_INBOUND"
chain11
```

コマンドを入力して、ソース10.0.0.0/16から来るIPv4トラフィックを受け入れます。 +

```
midonet> chain chain11 add rule ethertype 2048 src 10.0.0.0/16 type accept
chain11:rule0
```

行き先が10.0.5.0/24を持ったIPv4トラフィックを受け入れるためにコマンドを入力します。 +

```
midonet> chain chain11 add rule ethertype 2048 dst 10.0.5.0/24 type accept
chain11:rule1
```

1. コマンドを入力してIPv4トラフィック全て(これ以前のルールが持つ属性とマッチしなかったもの)をドロップします。

```
midonet> chain chain11 add rule ethertype 2048 pos 3 type drop
chain11:rule2
```

コマンドを入力して新しいルールチェーンに追加されたルールをリスト化します。
 . +

```
midonet> chain chain11 list rule
rule rule1 ethertype 2048 dst 10.0.5.0/24 proto 0 tos 0 pos 1 type accept
rule rule0 ethertype 2048 src 10.0.0.0/16 proto 0 tos 0 pos 2 type accept
rule rule3 ethertype 2048 proto 0 tos 0 pos 3 type drop
```

40

ポート上にインバウンドチェーン用のルールをリスト化

port1用にプレルーティングルールチェーンをリスト化するには、

次のコマンドを入力します。 . +

```
midonet> chain chain2 list rule
rule rule0 ethertype 2048 src !172.16.3.3 proto 0 tos 0 pos 1 type drop
rule rule1 hw-src !fa:16:3e:fb:19:07 proto 0 tos 0 pos 2 type drop
rule rule2 proto 0 tos 0 flow return-flow pos 3 type accept
rule rule3 proto 0 tos 0 pos 4 type jump jump-to chain4
rule rule4 ethertype !2054 proto 0 tos 0 pos 5 type drop
```

プレルーティングルールチェーンは、次の指示を含んでいます。

- ルール0は次のように述べています。各種パケットのうち、ethertype2048(IPv4)とはマッチするがソースIPアドレス172.16.3.3(これはVMのプライベートIPアドレス)とはマッチしないものについては、ドロップします。これらのパケットをドロップすることによって、ポートが模造IPアドレスつきのパケットをポートのVMが送信することを防止することができます。
- ルール1は次のように述べています。各種パケットのうち、そのハードウェアソースがリスト化されているソースMACアドレス(これはVMのMACアドレスのこと)とマッチしないものについては、そのパケットはドロップします。そうすることによって、VMが模造のMACアドレスつきのパケットを送信することを防止することができます。
- ルール2は次のように述べています。各種パケットのうちリターンフローとマッチしているもの(つまり、MidoNetが既に認知している接続に所属するパケットだということ)については、それらパケットを受け入れます。
- ルール3は次のように述べています。前記したルールとマッチした結果、ドロップはされずあるいは受け入れられることもなかったパケットについては、表示されたチェーン(chain4)にジャンプすることを許可します。
- ルール4は次のように述べています。各種パケットのうち、ethertype2054(ARPパケット)とはマッチしないものについては、それらパケットをドロップします。

オープンスタックセキュリティーグループのルールチェーンをリスト化します。

ルールチェーン全てをリスト化し、それからオープンスタックセキュリティーグループ用のルールチェーンを調べます。

ルールチェーン全てをリスト化しそして具体的にルールチェーンを調査する方法は次の通りです。

次のコマンドを入力します。 . +

```
midonet> list chain
chain chain5 name OS_SG_050593ed-56ad-44ef-8489-4052d02d99ff_INGRESS
chain chain0 name OS_PRE_ROUTING_5a151b0b-dea7-4918-bd17-876c1f7f5c64
chain chain1 name OS_POST_ROUTING_5a151b0b-dea7-4918-bd17-876c1f7f5c64
chain chain6 name OS_SG_01fce1b8-c277-4a37-a8cc-86732eea186d_INGRESS
chain chain4 name OS_SG_050593ed-56ad-44ef-8489-4052d02d99ff_EGRESS
chain chain7 name OS_SG_01fce1b8-c277-4a37-a8cc-86732eea186d_EGRESS
chain chain2 name OS_PORT_6f72342b-4947-432f-8d01-0cf4e4b8d049_INBOUND
chain chain3 name OS_PORT_6f72342b-4947-432f-8d01-0cf4e4b8d049_OUTBOUND
```


- ルールは全てethertype2048(IPv4)パケットとマッチします。
- ルールは全て、どのソースネットワーク(0.0.0.0/0)からのトラフィックともマッチします。
- ルール1を除くルールは全て、IPプロトコル6(TCP)のパケットとマッチしており、パケットを受け入れます。ルール1はICMP種別のパケットとマッチし、ICMP種別のパケットを受け入れます。
- ここまでですすでに述べたその他のマッチ事例の他にも、各種ルールは、自分がオープンスタックの中で定義したセキュリティグループルールに応じてパケットをマッチさせ受け入れますが、このことは、特に行き先を持ったパケットについてはありません。
- TCP port 5900 (VNC)
- TCP port 22 (SSH)
- TCP port 80 (HTTP)

テナント用にブリッジのリスト化

```
midonet> list bridgebridge bridge0 name demo-private-net state up
```

ブリッジ上にポートをリスト化

```
midonet> bridge bridge0 list port
port port0 device bridge0 state up
port port1 device bridge0 state up infilter chain2 outfilter chain3
port port2 device bridge0 state up peer router1:port1
```

[注記]

インフィルタ（プレルーティング）チェーンならびにアウトフィルタ（ポストルーティング）チェーン付きのポートはVMsに接続しています。ポート1は1つのVMに接続しています。

43

```
rule rule0 ethertype 2048 src 0.0.0.0/0 proto 6 tos 0 dst-port 5900 pos 1 type accept
rule rule1 ethertype 2048 src 0.0.0.0/0 proto 1 tos 0 pos 2 type accept
rule rule2 ethertype 2048 src 0.0.0.0/0 proto 6 tos 0 dst-port 22 pos 3 type accept
rule rule3 ethertype 2048 src 0.0.0.0/0 proto 6 tos 0 dst-port 80 pos 4 type accept
```

上記出力内容には、自分がオープンスタックの中に設定したセキュリティーグループを実装するために使用されたルールチェーンが表示されています。これらのルールには次のような指示内容が含まれています。

- ルールは全てethertype2048(IPv4)パケットとマッチします。
- ルールは全て、どのソースネットワーク(0.0.0.0/0)から来るトラフィックともマッチします。
- ルール1を除くいずれのルールも、IP protocol6(TCP)のパケットとマッチし、それらパケットを受け入れます。ルール1はICMP種別のパケットとマッチし、ICMP種別のパケットを受け入れます。
- すでに述べた他のマッチ事例の他にも、各種ルールは、自分がオープンスタックの中で定義をしたセキュリティーグループルールに応じてパケットをマッチさせ受け入れます。この点は特に行き先を持ったパケットについて当てはまります。
 - TCP port 5900 (VNC)
 - TCP port 22 (SSH)
 - TCP port 80 (HTTP)

假定事項

下記にある例については、以下のようなネットワークコンディションがあることを仮定します。

- ・テナントのルーターの名称を”tenant-router”とします。
- ・プライベートネットワークのアドレスを（172.16.3.0/24）とします。
- ・パブリックネットワークのアドレスを（198.51.100.0/24）とします。
- ・プライベートIPアドレスが（172.16.3.3）とパブリック（フローティング）IPアドレス（198.51.100.3）のVMがあります。 == プレルーティングルールを閲覧

現在のテナント上のルーター、また、ルーターのルールチェーン情報をリストアップするために、以下のコマンドを入力します。

```
midonet> list router
router router0 name tenant-router state up infiltr chain0 outfilter chain1
```

上記のアウトプットにあるように、"chain0" はルーターのプレルーティング（インフィルタ）ルールチェーンで、"chain1" はポストルーティング（アウトフィルタ）ルールチェーンをあらわしています。

ルーターのプレルーティングルールチェーンについての情報をリストアップするためには、以下のコマンドを入力します。

```
midonet> chain chain0 list rule
rule rule0 dst 198.51.100.3 proto 0 tos 0 in-ports router0:port0 pos 1 type dnat action
  accept target 172.16.3.3
rule rule1 dst 198.51.100.2 proto 0 tos 0 in-ports router0:port0 pos 2 type rev_snat
  action accept
```

テナントのルーター上のプレルーティングルールチェーン” rule0” は以下のインストラクションを含みます。

- VMに連携しているフローティングIPアドレスの行き先が (198.51.100.3) のパケット
- 行き先のIPアドレスをVMのフローティングIPアドレス(198.51.100.3)からVMのプライベートIPアドレス(172.16.3.3)へ変更するために行き先NAT(DNAT)転換を行います。

[ポストルーティングルールを閲覧](#)

テナントのルーター上のポストルーティングルールをリストアップするには、以下のコマンドを入力します。

```
midonet> chain chain1 list rule
rule rule0 src 172.16.3.3 proto 0 tos 0 out-ports router1:port0 pos 1 type snat action
  accept target 198.51.100.3
rule rule1 proto 0 tos 0 out-ports router1:port0 pos 2 type snat action accept target 198.
51 100 2:1--1
```

テナントルーター上のポストラーティングルールの” rule0” は以下のインストラクションを含みます。 * ソースIPアドレス (172.16.3.3) からのパケット (VMのプライベートIPアドレス) です。

SNAT, DNAT, REV DNATの設定

上記にあるDNAT設定のためのルールチェーンの作成方法が記載されています。

1. 新たなルールチェーンを作成します。

```
midonet> chain create name "dnat-test"
chain10
```

2. ルーターポート上の行き先が (198.51.100.4) のトラフィックを承認し、行き先を (10.100.1.150) にこれに転換するルールを作成します。

```
midonet> chain chain10 add rule dst 198.51.100.4 in-ports router1:port0 pos 1 type dnat
action accept target 10.100.1.150
chain10:rule2
```

3. ルーターのゲートウェイからパブリックネットワークへの行き先をもつトラフィックを承認するルールを作成します。そして、パブリックネットワークアドレスからプライベートネットワークアドレスへとアドレスの逆転換を行います。

```
midonet> chain chain10 add rule dst 198.51.100.2 in-ports router0:port0 pos 2 type
rev_snat action accept
chain10:rule3
```

4. 以下を確認するルールをリストアップします。

```
midonet> chain chain10 list rule
rule rule2 dst 198.51.100.4 proto 0 tos 0 in-ports router1:port0 pos 1 type dnat action
accept target 10.100.1.150
rule rule3 dst 198.51.100.2 proto 0 tos 0 in-ports router0:port0 pos 2 type rev_snat
action accept
```

SNAT例

ルーターポート上のSNATを設定するために、MidoNet CLIをどのように使用するかの例を記載しています。

例えば、プライベートIPアドレスとパブリックフローティングIPアドレスをもつVMのようなネットワークデバイス用にSNATを設定します。それらのIPアドレスは自身のもつローカルネットワークの外へとデータを転送します。

本例のルールチェーンについての仮定事項が以下に記載されています。 * パブリックサブネット(198.51.100.0/24)とプライベートサブネット(10.0.0.0/24)の二つサブネットがあります。

- パブリックIPアドレス (198.51.100.4) とプライベートIPアドレス (10.100.1.150) でテナントルーターに接続されているバーチャルデバイスがあります。
- VMのプライベートIPアドレスをVMのパブリックIPアドレスへとトラフィック転換を行います。

上記にあるSNAT設定のためのルールチェーンを作成します。

1. 新たなルールチェーンを作成します。

```
midonet> chain create name "snat-test"
```

T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

- T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

- T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

第9章 レイヤ4のロードバランシング

目次

ロードバランサーの設定	51
スティッキーソース IP	53
ヘルスマニター	54

MidoNetのロードバランサーはレイヤ4 (L4)のロードバランシングを提供します。フローティングIPアドレスからプライベートIPアドレスにトラフィック（ロード）をバランスしたいテナントに対応する場合などが典型的な事例です。

一般的にトラフィックは、外部もしくはパブリックのルータブルアドレス（例えば、サービスを使う世界中のユーザーです）から仮想IPアドレス（これは通常はパブリックのフローティングIPアドレスですが、VMの替わりにロードバランサーにアサインされます）にやってきます。そして、クラウド上で数多くのプライベートIPアドレスにロードバランスされます。MidoNetの場合は、これらのバックエンドのサーバーはVMです。ロードバランサーは接続を打ち切らず、ディスティネーションIPにトランスレートするので、あまりトランスパレントではありません。

設定のオーバービュー

設定の一環として、トラフィックがロードバランスされているバックエンドサーバーのプール(VM)を定義することができます。プールメンバーは通常プライベートIPアドレスを持っています。VMの配置は柔軟ですが、ロードバランサーがどこに設定されるかは考慮する必要があります。一般ルールとして、VMプライベートアドレスは、ロードバランサーがアタッチされているルーターから到達される*必要があります*。したがって、プールメンバーは、

- 全てルーターのシングルサブネットに全て属するか、もしくは
- ルーターの複数のサブネットに配置されます

最終的に、ロードバランサーはリクエストソースアドレスを修正せずに残すので、ロードバランサーはリターントラフィックのパスに配置される必要があることに留意してください。フォワードバケットに適用しているVIP→back-end-IPトランスレーションという流れを反転する機会をロードバランサーに与えずにリターントラフィックは、リクエストソースに反映されます。

ヘルスマモニタリング

それに加えて、バックエンドサーバーのチェックを行う為のヘルスモニターの設定ができます。ヘルスモニターはプールから、健全でないノードを自動的に取り除き、健全に戻ったら追加します。

MidoNetロードバランサーの制約

MidoNetはロードバランサーを設定するために、Neutron APIを使います (https://wiki.openstack.org/wiki/Neutron/LBaaS/API_1.0でドキュメント化されています) しかし、すべてのNeutron LBaaSフィーチャーがMidoNetでサポートされているわけではありません。

- ・ L7のロードバランシングはサポートされていません。

-

このルールは、やってくるトラフィック(src 0.0.0.0/0)をプロバイダ
 ルーターにマッチします。これは、プロバイダルーター203.0.113.2
 (dst 203.0.113.2/32)のVIPに送られて、プロバイダルーターポート
 3(router1:port3)に送ります。

スティッキーソース IP

多くの場合、セッションの記録を取る為にロードバランサーを使います。これを行う為に、MidoNetロードバランサーはスティッキーソースのIPアドレスパーシステンスを提供します。

仮想IP(VIP)を設定する時に、パケットのソースIPアドレスは、ディスティネーションサーバーを決定する時に使われます。そして、同じソースサーバーからくるその後のトラフィックは、同じサーバーに送られます。

セッションパーシステンスの例

下記の例は、ロードバランサーを設定する為にMidoNet CLIをどうやって使うかを示したものです。示しているように、VIPはSOURCE_IPに対してパーシステンスなセットです。

```
midonet> load-balancer create
lb0
midonet> router router0 set load balancer lb0
midonet> load-balancer lb0 create pool lb-method ROUND_ROBIN
b0:pool0
midonet> load-balancer lb0 pool pool0 create member address 192.168.100.1 protocol-port 80
b0:pool0:pm0
midonet> load-balancer lb0 pool pool0 list vip
midonet> load-balancer lb0 pool pool0 create vip address 192.168.0.1 persistence SOURCE_IP
protocol-port 8080
lb0:pool0:vip0
midonet> router router1 add route dst 192.168.0.1/32 src 0.0.0.0/0 type normal port
router1:port0
router1:route11
```



注記

ポート8080は参考例です。ロードバランシングトラフィックのためにポートを使うには、まずそれが、他で使われていないかを確認する必要があります。



重要

- VIPのスティッキーソースIPアドレスモードを切り替えon/offする場合、そのVIPを使う既存の接続はドロップします。
- スティッキーソースIPアドレスモードで、プールメンバーを利用不可能にする場合は、そのメンバーに対してバランスしている接続はドロップされます。
- スティックソースIPアドレスモードでない時に、プールメンバーを利用不可能にしたら、そのメンバーをバランスしている既存の接続は完了

することができます。しかし、そのメンバーに対して、新しいコネクションは送られません。 ＊スティッキネスがアクティブな状態にあるのは一日です。セッションが一日以上アクティブになったら、スティッキネスは無くなってしまい、後続のトラフィックが通常通りにロードバランスされます。

ヘルスマニター

ヘルスモニタリングは、プールメンバーが“生きているか”をチェックするアクティビティです。つまり、HTTP, TCP, UDP, もしくは ICMPでの接続性が、そのノードで確立しているかを確認します。

MidoNetのケースでは、TCPでの接続性のみをチェックします。ヘルスマonitoringはパケットをプールメンバーに送って、返信を受け取ることを確認します。 何度かリトライした後に、ある時間以内にパケットをプールメンバーが返信したら、そのノードはACTIVEとして考慮されます。従って、ヘルスマonitorは以下の三つの変数によって成り立ちます：

- `max_retries`: ヘルスモニターがノードをINACTIVEとして判断する前に、ヘルスモニターが返信が無い状態で何度パケットをプールメンバーに送ったか
- `delay`: ヘルスモニターからプールメンバーに対して、パケットを送送する間隔の時間です
- `timeout`: 接続が確立されてから追加のタイムアウトです

ヘルスマニターは、アサインされた全てのプールメンバーの現状のステータスのトラッキングを行います。ロードバランシングの決定は、プールメンバーが“生きている”かがベースになります。

HAProxy 設定

レイヤー4 ロードバランサーを使う時は、バックエンドサーバーのチェックを行う為にヘルスモニターを設定します。

一度に、一つのホストしか全てのヘルスモニターを走らせることしかできません。もし、ホストが何らかの理由でダウンしてしまったら、別のホストが選出され、HAProxyインスタンスを起動します。HAProxyインスタンスはMidoNetエージェントによって管理されており、設定は必要ありません。しかし、HAProxyインスタンスを潜在的に保持するホストを選択する必要があります。

ヘルスモニタリングのためのMidoNet Agentホストを有効にするには、そのホストの `health monitor enable` プロパティが `true` に設定する必要があります。

ホストはヘルスモニタリングが有効になっているかどうかを確認するには、次のコマンドを実行します。

```
$ mn-conf get agent.haproxy health monitor.health monitor enable
```

特定のホストのヘルスマモニタリングを切り替えるには、そのホスト上で次のコマンドを使用します：

```
$ echo "agent.haproxy health monitor.health monitor enable : true" | mn-conf set -h local
```

```
$ echo "agent.haproxy health monitor.health monitor enable : false" | mn-conf set -h local
```

それに加えて、HAProxyインスタンスを走らせているホストは、"nogroup"と呼ばれるグループと"nobody"と呼ばれるユーザーを持つ必要があります。そうでないな

T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

- T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

目次

マルチキャストフレームにおけるような特定のL2フレームを除外するためにL2アドレスマスクINGを使用することができます。これによって、必要なルール数を減らすことのできる特定マスクとの単一ルールを加えることができます。

- OpenStackのセキュリティグループルールはL2フィールドとは合致しないため、MidoNetのAPIやCLIに直接アクセス、または使用されるお客様のために本機能が提供されました。
- CLI内のhw-dst-maskアトリビュート用の値を含まない古いルールは、デフォルト設定されたフィールドまたはアトリビュート用の値をもっていると解釈されます。デフォルト値はffff.ffff.ffffと記載されます。本機能を実装するためのルールチェーンについての情報（“hw-src-mask”や“hw-dst-mask”アトリビュートについての情報も含まれます。）は、[7章ルールチェーン \[31\]](#)を参照します。

L2アドレスマスキュールチェーン例

1. Create a chain (this creates chain alias, "chain0" in the example, pointing to the chain created): チェーンを作成します。(作成されたチェーンにポイントされているチェーンエイリアスを作成します。エイリアスの例としては、"chain0" が挙げられています。)

2. トラフィックをドロップするルールをチェーンに追加します。ルールは、src (source) MAC not starting with 12:34:56となります。

例

hw-dst-maskの使い方の例をここに挙げています。

特定のL2バーチャルネットワーク（ブリッジ）上の全てのマルチキャストトラフィックをブロック（ドロップ）します。MACアドレスの最初のオクテット（最も重要）の8番目のビット（最も重要でない）がマルチキャストなら”1”でユニキャストなら”0”になります。

MACブロードキャストアドレスの全てのオクテットが”FF”に設定されている場合、ARPリクエストなどのブロードキャストパケットをドロップしないことをお勧めします。従ってそのような場合は、以下の二つのルールをバーチャルブリッジのプレブリッジルールチェーンに追加します。 * ポジション1では、もし”hw-dst”が”ff:ff:ff:ff:ff:ff”の場合、承認します。

T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

- T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

目次

- L2/L3ファイアウォールを書き込んでいる場合、IPフラグメントの影響をうけないようにすることができます：IPフラグメントの特殊なハンドリングは必要ありません。（L3 NATがIPフラグメントを正確にハンドリングします。）
- L4ファイアウォールを書き込んでいる場合、IPフラグメントの特殊なハンドリングを指定します。

- ・ヘッダー = フラグメントされていないパケットとヘッダーフラグメントが一致しているコンディションのことです。ヘッダーはフラグメントされていないパケット、またはフラグメントされているパケットの最初のフラグメント（ヘッダーフラグメント）を指します。ヘッダーフラグメントはIPv4フラグメントオフセットのフィールドがゼロに設定されている唯一のものです。ヘッダーはルールのコンディションがL4フィールドと一致する場合、またはルールがダイナミック（それは、ポートが変更している）DNATもしくはSNATの場合、唯一の許容される値です。
- ・ノンヘッダー = ノンヘッダーフラグメントのみと一致しているコンディションのことです。ノンヘッダーは、二つ目またはそれ以降のフラグメントのことを指します。
- ・エニー = フラグメントされている、または、されていないパケットと一致するコンディションのことです。
- ・アンフラグメント = フラグメントされていないパケットと一致するコンディションのことです。



59



シングルL4のフローは最大で2種類生成されます。一つ目はノンヘッダーフラグメントを処理するもので、もう一つはその他のパケットを処理するものです。

フラグメントされたパケットルールチェーン生成例

```
midonet> chain chain0 add rule ethertype 2048 src 0.0.0.0/0 dst 0.0.0.0/0 in-ports
router2:port0 pos 2 type drop
```

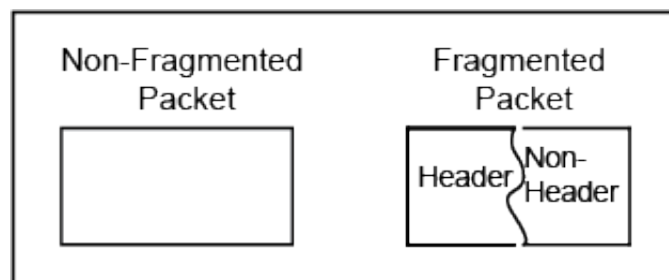
```
midonet> chain chain0 list rule
rule rule0 ethertype 2048 src 0.0.0.0/0 proto 6 tos 0 dst-port 80 in-ports router2:port0
pos 1 type accept
rule rule1 ethertype 2048 src 0.0.0.0/0 dst 0.0.0.0/0 proto 0 tos 0 in-ports router2:port0
pos 2 type drop
```

上記のルールチェーンにより、MidoNetは以下にあるように行き先になるTCPポート80としてフラグメントされたパケットを処理します：

- TCPヘッダーを含む、最初半分のパケットはポジション 1 でのルールに到達し承諾されます。
- 一方、残り半分の行き先になるポートをもたないフラグメントはポジション 1 ルールに到達し、ルールのコンディションと一致しないためドロップされます。これはフラグメントされたパケットがウェブサーバーに到達しないことを意味します。

例2 ファイヤーウォールはフラグメントされたパケットをアドレス指定します。

この問題を特定するために、MidoNetはフラグメントされたパケットを処理するメカニズムを提供しています。下記の例にあるように、このメカニズムによりフラグメントされたパケットをそれぞれの行き先に到達させることができます。下記イメージに、ヘッダーとノンヘッダーの2部を含むフラグメントされていないパケットとフラグメントされているパケットの全体像が描写されています。



フラグメントされていないパケットとフラグメントされているパケット

本例には以下のパッケージが含まれています。

- ・ 行き先がTCPポート80のフラグメントされていないパケット
- ・ 行き先がTCPポート80のフラグメントされているパケット
- ・ 行き先がTCPポート72のフラグメントされていないパケット
- ・ 行き先がTCPポート72のフラグメントされているパケット



- パケット 1 とポジション 1 ルールが一致すると承諾されます。
- パケット 2 のヘッダー部分がポジション 1 ルールと一致する場合承諾されます；行き先のないノンヘッダーフラグメントはルールと一致しないのでドロップされます。
- パケット 3 の行き先がポジション 1 ルールと一致しない場合、パケット 4 のヘッダー部分と同様にドロップされます。パケット 4 のノンヘッダー部分に行き先の情報がない場合もドロップされます。

はじめの目的は、ヘッダーを含むフラグメントされているパケット部分を承諾することです。これをするためにポジション1で同様のルールを生成します。そして、TCP/UDPヘッダーを含む全てのパケットをドロップするためにポジション2にて新たなルールを追加します。

- ・ ポジション 1 ルール
 - ・ デフォルト設定により、このルールはフラグメントされていないパケットとヘッダーフラグメントを一致させます。
 - ・ protocol=TCP、destination=80を含むin-ports=router2:port0からのパケットを承諾します。

- ・ **ポジション 2ルール**

```
midonet> chain chain18 add rule ethertype 2048 src 0.0.0.0/0 in-ports router2:port0
fragment-policy header pos 2 type drop
```

```
midonet> chain chain18 add rule ethertype 2048 src 0.0.0.0/0 in-ports router2:port0
dst 0.0.0.0/0 pos 3 type accept
```

ポート72行きのパケットからはじまる上記にあるパケットが、新たに設定されたルールチェーンをどのように進行するかを参照します。

- パケット3の行き先はポート72であってポート80とは異なります。 よってポジション1ルールと一致しないため、ポジション2ルールに進みます。
- パケット3はTCPヘッダーを含みます。 ポジション2ルールと一致するためにドロップされます。
- パケット4のヘッダーフラグメントはポート72への行き先を含むため、ポジション1ルールと一致せず、ポジション2ルールへと進みます。
- このフラグメントはTCPヘッダーを含み、 ポジション2ルールと一致するためドロップされます。
- パケット4のノンヘッダーフラグメントはヘッダーを含まない（つまり行き先の情報がない）ため、ポジション1ルールと一致せずポジション2ルールへと進みます。
- このノンヘッダーパケットフラグメントはTCP/UDPヘッダーを含まないためポジション2ルールと一致せず、 ポジション3ルールへと進みます。
- ポジション3ルールでは、ここに到達する全てのパケットフラグメントを承諾します。 関連するヘッダー情報がないために、再構成されずにアプリケーションに送られ、いずれドロップされます。

パケット 1 と 2 を参照します。

- パケット 1 の行き先がTCPポート80でポジション 1 ルールと一致するため承諾されます。
- パケット 2 では、TCPポート80の行き先を含むヘッダーをもつパケットフラグメントはポジション 1 ルールと一致するため承諾されます。
- ノンヘッダーパケットフラグメントをもつパケット 2 はヘッダーを持たず、ポジション 1 ルールと一致しないためポジション 2 ルールへと進みます。
- このノンヘッダーパケットフラグメントはTCP/UDPヘッダーを含まないためポジション 2 ルールと一致せずドロップされ、ポジション 3 ルールへと進みます。
- ポジション 3 ルールでは、全てのパケットを承諾するため、このパケットフラグメントも承諾されます。

この変更によってノンヘッダーフラグメントがポジション1と2ルールを通過することができ、ルールチェーンを承諾して終了することができます。また、この変更によりファイヤーウォールは全てのノンヘッダーフラグメントを通過させますが、リスクレベルが許容範囲にあると判断され、不適切なHTTPフローの修正を行います。該当

T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT

設定

リソースプロテクションの無効化

目次

当チャプターはそれぞれのサービスに適用可能な主要メトリクスについて、また、またMidoNetデプロイメントパッケージに与えられたスクリプトに基づき、Muninに基づいた基本的なモニタリングインフラ設定をするプロシージャについて説明します。

測定

概要

注意: 測定データは時系列データベース上のモニタリングレイヤーによってポーリング及び保存されるようにされている。なので、エージェントは集めた測定データを保持しないので、エージェントがリブートした際には測定値はゼロにリセットされます。全てのメーターデータコレクションレイヤーはこの影響を考慮すべきであり、またカウンターのリセットを検出すべきです。

メーターのクエリ

エージェントはJMXよりメーターを発行し、コマンドラインツールである `mm-meter` はメーター値をリスト化、フェッチそしてモニタリングをするためにJMXインターフェースを使います。

JMXインターフェース上のコードの例の参照には、以下をご参照ください [the code of mm-meter itself](#)

メーターを`mm-meter`でクエリするのは非常に簡単です。

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

Subcommand: list - list all active meters
--help              Show help message
Subcommand: get
-n, --meter-name <arg> name of the meter
--help              Show help message

trailing arguments:
delay (not required)  delay between updates, in seconds. If no delay is
                      specified, only one report is printed. (default = 0)
count (not required) number of updates, defaults to infinity
                      (default = 2147483647)
```

`list`コマンドはこのエージェントが知りうる全てのメーターのリストを表示します。

```
$ mm-meter list
meters:user:port0-on-the-bridge
meters:port:rx:cf453c9d-94c4-4c27-ba32-529b7cbacf1d
meters:device:845a54bf-b702-4dc2-8958-bbe7156bc4ef
meters:port:tx:cf453c9d-94c4-4c27-ba32-529b7cbacf1d
meters:port:tx:9182485b-8f86-462d-a8be-62586060eeb9
meters:device:f0d1f093-2de7-49a1-a5ec-898f94769e34
meters:device:9182485b-8f86-462d-a8be-62586060eeb9
meters:port:rx:9182485b-8f86-462d-a8be-62586060eeb9
meters:device:cf453c9d-94c4-4c27-ba32-529b7cbacf1d
```

そして`get`コマンドはcurrent, local countersをメーターに表示します。これにより遅れが生じますがその場合には定期的にメーターをポーリングして超過分を表示します。

```
$ mm-meter get -n meters:port:rx:cf453c9d-94c4-4c27-ba32-529b7cbacf1d 10
  packets      bytes
  568935      4215888475
    0          0
    0          0
    23         5834
    0          0
```

カスタムメーターの作成

運用者は仮想ネットワークトラフィックのカスタムスライスを測定したい場合があります。これは、仮想トポロジー内のひとつもしくはいくつかのチェーンルールを使ってそのスライスをマッチすることで可能です。フローが自然に与えるメーターに加えて、チェーンルール内の`meterName`プロパティは自らの値により参照されたメーターにマッチングフローをアサインします。

グループとして現れます。このガイドの後にあるセクションで、サービス特有のグラフをどのようにして加えるかの説明をしています。

スレーブノード

*munin-node*と*munin-plugin-extra*パッケージをインストールしてください。
また、jmx2muninをGitHubリポジトリよりインストールしてください。/etc/munin/
munin-node.conf.sampleを再利用して単純にマスターのIPアドレスをアップデート
することができます。

プラグイン

Muninはディスク、メモリ、CPUやその他のためのメトリクスを示すいくつかのデフォルトプラグインを提供します。現時点ではこれらは使用しない方がいいかもしれませんが、それぞれのスレーブの/etc/munin/pluginsにあるシンボリックリンクを削除することによって、これらを簡単に取り除くことができます。

MidoNetプラグイン

MidoNetサブシステムのためにメトリクスをフェッチするには、関連するスレーブに新しいプラグインを加えることとのみが必要です。ほとんど全ての作業ができる自動化されたスクリプトがあります。MidoNetデプロイメントリポジトリを現在のディレクトリでコピーしていると想定して、それぞれのスレーブにて以下のコマンドを実行するのみです。

```
sudo ./install_plugins.sh
sudo service munin-node restart
```

RedHat Enterprise Linux 7にて以下を使用してください。

```
sudo systemctl restart munin-node
```

これは全ての利用可能なプラグインをインストールし、Munin設定をアップデートします。

マスターへは何もする必要はありません。Muninフェッチプロセスが実行されると、データのフェッチが始まります。待つのが難しい場合は、スレーブよりデータを引き出すMuninクーロンジョブを始動させる、以下のコマンドをマスターで実行することもできます。

```
sudo -u munin munin-cron
```

Zabbixを使用したモニタリング

このセクションは、イベントモニタリングのためMidoNetとZabbixを統合する方法を説明します。

Zabbixは、システムアドミニストレータがネットワークをモニターするためのモニタリングツールとして人気があります。Zabbix使用の詳細に関しては、<https://www.zabbix.com/>内のドキュメンテーションをご覧ください。

Zabbix使用の基本ステップ

Zabbixでは、ファイルシステム内に保管されたログファイルをモニターすることができます。それぞれのノードにつきZabbixエージェントを実行し、ファイルロケーションや、アップデートの頻度などのログモニタリング設定をする必要があります。

- Zabbixをそれぞれのゲートウェイで実行しホストを算出します。

- ログファイルモニタリングを設定します; これには下記のタスクを含みます。
- エージェントパラメーターを検証します。
- 項目を設定します(項目とはホストより読み出したいメトリックデータの一部です。)
- マッチするための通常の式とともに、モニターするためのログファイルへのパスを規定します。
- 規定したイベントが起こった際にユーザーに知らせるためのトリガーの設定をします。

詳細に関しては https://www.zabbix.com/documentation/2.0/manual/config/items/itemtypes/log_items をご参照ください。

ネットワークステイトデータベースモニタリング

ネットワークステイトデータベースはCassandraインスタンスとZookeeperインスタンスによりデプロイされます。この両インスタンスはJMXバインディングを提供しています。

MidoNetに提供される設定は、我々の利用するケースにもっとも関係のあるサブセットのみ使います。下記セクションの詳細に、MidoNetのデプロイメントスクリプトにより設定されたメトリクスについての追加情報と、注意すべき点についての説明があります。

Cassandra

デフォルトで、Cassandraはその全てのノードからJMXサービスのためポート7199を使い、包括的な見解のためjコンソールを使って接続することができます。

加えて、Cassandra独自のノードツールユーティリティは与えられたノードにおいて、cfstatsやtfpstatsのような、有益な統計値がキースペース、テーブル、コラムファミリー等へのアクセスができるようになるコマンドを提供します。

モニタリングへの豊富なレファレンスについては、公式ドキュメンテーションをご参照ください(<http://www.datastax.com/>にて "monitoring a Cassandra cluster" を検索してください)。

下記はMuniMidoNetデプロイメントリポジトリ内で与えられたMunin設定の例から作られたグラフの説明になります。このグラフがCassandra JMXサービスのサブセットから作られたものになります。利用可能なグラフは、

キャッシュ要求 vs. ヒット

これは自称で、キャッシュヒットがリクエストにできる限り近づくことが理想です。デフォルトではMidoNet CassandraノードはPartition Key Cacheだけを可能にし、Row Cacheはしませんので、これらが0のままでいるのは普通なことであるということに注意してください。MidoNetにとって、Partition Key Cacheは実際上Row Key Cacheにとっても似ているはずです。なぜなら我々のコラムファミリー (CF) は一つしか列を持っておらず、それゆえ行はいくつかのSSTablesには広がっていないからです。

コンパクション

これは圧縮されたバイトの数を示しています。典型的な作業負荷は、小さなコンパクションが実行された時通常の小さなスパイクを表示し、また大きなコンパクションが実行された時頻度の低い大きなスパイクを表示します。大量のコンパクションはクラスターの容量を増やす必要があることを表します。

内部タスク

これらは内部のCassandraタスクです。一番重要なのは、

- **Gossip:** MidoNetのCassandraノードはGossip (Gossipの中にて、ピアの間で状態情報が行き来されます) の中でかなり多くの時間を使うことが予想されます。
- **MemTable Post Flusher:** memtableはコミットログにかかることを待っているものを洗い流します。これらはできる限り低くあるべきでとどまるべきではありません。
- **Hinted Handoff tasks:** これらのタスクが現れるときは、レプリカが利用不可能ということが検出されたことを示します。なので、レプリカが利用可能になるまでの間、レプリカではないノードが一時的にデータを保管する必要があります。 頻繁なHinted Handoffスパイクはクラスターからノードがパーテーションで区切られていることを示唆しているかもしれません。
- **反エントロピースパイク:** データの不一致が検出されまた解決されたことを示します。
- **ストリームアクティビティ:** 他のノードよりデータを転送するもしくは要求することを含みます。これらは頻繁には起こらず、また容量をとらないことが理想です。

Messaging サービスタスク

これらはそれぞれのピアノードにて受け取られまた答えられたタスクです。全てのピアに均等なディストリビューションが期待されます。

NAT Column Familyレイテンシ

NATマッピングキャッシュの読み書きレイテンシについて知らせるキーマトリックです。読みの場合特に高いレイテンシは問題となります。なぜなら、NATルールが適用される仮想ルーターを横断するトラフィックに高いレイテンシを起こすからです。Cassandra自身の保証により、書きレイテンシは低くなると考えられます。高い応答レベルはレイテンシに非常に大きな影響を与えるということにご注意ください（ノードはレプリカよりACKを読み出し受け取らなくてはなりません）。特に、なくなってしまったキャッシュ内などにおいて、レイテンシ内のスパイクはコンパクションのようなイベントと相互に関係していることがあります。コンパクションのせいで、Cassandraは高いI/Oロードの間、データをフェッチするためにディスクに行く必要があるからです。

NATコラムファミリーMemtable

データサイズとコラムカウントを示します。これはインメモリーデータです。マッピングの生存時間 (TTLs) が期限切れになった後にほとんどのデータが期限切れになるので、シーソーパターンを予測してください。

NATコラムファミリーディスク利用

キーが表示されていないときにキャッシュに格納するために保管するために使われた Bloom filterのために使われたものを含む、全般的なディスク利用を表します。

ZooKeeper

install_plugins.shスクリプトは、ZooKeeperの露出したJMXマトリクスからグラフを作り出す、Muninプラグインと設定ファイルもインストールします。

それぞれのノードにおいて、ZooKeeperのレプリカ番号を示す必要があります;スクリプトはどのようにしてこの作業を行うかのわかりやすいガイダンスを与えてくれます。

ZooKeeperの統計データはMidoStorageグループ”zookeeper”カテゴリー内で見つけることができます。ZooKeeperはリーダー/フォロワーのため、メトリクスを別々のMBeansに分類します。それぞれのノードは同じ値を2回レポートします、一つはフォロワーロール内で、もう一つはリーダーロール内です。与えられたノードはロールを変えることがありえるということをふまえてください(例えば、もしリーダーがシャットダウンしたら、フォロワーノードがリーダーにとってかわることがあり得ます)。これらのイベントは簡単に見ることができます。例えば、”フォロワーとしての接続カウント”内のラインが急に無効になり、”リーダーとしての接続カウント”内に別のラインが現れます。

以下は、MidoNetデプロイメントリポジトリ内で与えられたMunin設定の例からのグラフの説明です。グラフはZooKeeper JMXサービスのサブセットより作られました。利用可能なグラフは、

コネクションカウント(フォロワー/リーダーとして)

これらの二つのグラフは任意の時点での、ロールにおけるこのノードへのライブ接続の数を表示しています。

メモリーデータツリーにて(フォロワー/リーダーとして)

データノードとウォッチカウント両方の、インメモリーノードデータベースのサイズを表示します。

レイテンシ (フォロワー/リーダーとして)

接続内で経験されたレイテンシ平均および最大値を表示します。

Packet Count (フォロワー/リーダーとして)

任意の時点において、ロール内でノードにより送信/受信されたパケットのカウンタを表示します。

定数サイズ

リーダーの選出に合意しているノードの数についてのそれぞれのノードの観点を表示します。

ZooKeeperはそれぞれの特定の接続についての情報も見せます。これはトラブルシューティングの際に役立つかもしれません。 jconsole (<http://www.oracle.com/technetwork/java/index.html>にて ”jconsole” と検索をして情報を参照してください)を使って、以下が可能となります。

1. ポート9199で、任意のZooKeeperノードに接続します。
2. org.apache.ZooKeeperService, ReplicatedServer_idXへナビゲートします。
3. 望ましいレプリカを選びます。

- 77

ルーター

Logger	org.midonet.event.topology.Router.CREATE
Message	CREATE routerId={0}, data={1}.
Level	INFO
Explanation	routerId={0}のルーターが作成されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Router.UPDATE
Message	UPDATE routerId={0}, data={1}.
Level	INFO
Explanation	routerId={0}のルーターは{1}にアップデートされました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Router.DELETE
Message	DELETE routerId={0}.
Level	INFO
Explanation	routerId={0}のルーターが削除されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Router.ROUTE_CREATE
Message	ROUTE_CREATE routerId={0}, data={1}.
Level	INFO
Explanation	routerId={0}内にRoute={1}が作成されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Router.ROUTE_DELETE
Message	ROUTE_DELETE routerId={0}, routeId={1}.
Level	INFO
Explanation	routerId={0}内にてrouteId={1}が削除されました。
Corrective Action	N/A

ブリッジ

Logger	org.midonet.event.topology.Bridge.CREATE
Message	CREATE bridgeId={0}, data={1}.
Level	INFO
Explanation	bridgeId={0}のブリッジが作成されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Bridge.UPDATE
Message	UPDATE bridgeId={0}, data={1}.
Level	INFO
Explanation	bridgeId={0}のブリッジが{1}にアップデートされました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Bridge.DELETE
Message	DELETE bridgeId={0}.
Level	INFO
Explanation	bridgeId={0}のブリッジが削除されました。

ポート

Corrective Action	N/A
-------------------	-----

Logger	org.midonet.event.topology.Port.CREATE
Message	CREATE portId={0}, data={1}.
Level	INFO
Explanation	portId={0}のポートが作成されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Port.UPDATE
Message	UPDATE portId={0}, data={1}.
Level	INFO
Explanation	P portId={0}のポートは{1}にアップデートされました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Port.DELETE
Message	DELETE portId={0}.
Level	INFO
Explanation	portId={0}のポートが削除されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Port.LINK
Message	LINK portId={0}, data={1}.
Level	INFO
Explanation	portId={0}のポートがリンクされました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Port.UNLINK
Message	UNLINK portId={0}, data={1}.
Level	INFO
Explanation	portId={0}のポートがリンクをはずされました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Port.BIND
Message	BIND portId={0}, data={1}.
Level	INFO
Explanation	portId={0}のポートがバインドされました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Port.UNBIND
Message	UNBIND portId={0}.
Level	INFO
Explanation	portId={0}のポートがバインドを外されました。
Corrective Action	N/A

チェーン

Logger	org.midonet.event.topology.Chain.CREATE
Message	CREATE chainId={0}, data={1}.
Level	INFO

Logger	org.midonet.event.topology.Chain.DELETE
Message	DELETE chainId={0}.
Level	INFO
Explanation	chainId={0}のチェーンが削除されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Rule.DELETE
Message	DELETE ruleId={0}.
Level	INFO
Explanation	ruleId={0}のルールが削除されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.TunnelZone.MEMBER_CREATE
Message	MEMBER_CREATE tunnelZoneId={0}, data={1}.
Level	INFO
Explanation	TunnelZone member={1}がtunnelZoneId={0}に追加されました。
Corrective Action	N/A

BGP

Logger	org.midonet.event.topology.TunnelZone.MEMBER_DELETE
Message	MEMBER_DELETE tunnelZoneId={0}, data={1}.
Level	INFO
Explanation	TunnelZone member={1}がtunnelZoneId={0}より削除されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Bgp.CREATE
Message	CREATE bgpId={0}, data={1}.
Level	INFO
Explanation	bgpId={0}のBGPが作成されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Bgp.UPDATE
Message	UPDATE bgpId={0}, data={1}.
Level	INFO
Explanation	bgpId={0}のBGPが{1}にアップデートされました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Bgp.DELETE
Message	DELETE bgpId={0}.
Level	INFO
Explanation	bgpId={0}のBGPが削除されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Bgp.ROUTE_CREATE
Message	ROUTE_CREATE bgpId={0}, data={1}.
Level	INFO
Explanation	Route={1}がbgpId={0}に加えられました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Bgp.ROUTE_DELETE
Message	ROUTE_DELETE bgpId={0}, data={1}.
Level	INFO
Explanation	Route={1}がbgpId={0}より削除されました。
Corrective Action	N/A

ロードバランサー

Logger	org.midonet.event.topology.LoadBalancer.CREATE
Message	CREATE loadBalancerId={0}, data={1}.
Level	INFO
Explanation	loadBalancerId={0}のロードバランサーが作成されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.LoadBalancer.UPDATE
Message	UPDATE loadBalancerId={0}, data={1}.
Level	INFO
Explanation	loadBalancerId={0}のロードバランサーが{1}にアップデートされました。

VIP

Corrective Action	N/A
Logger	org.midonet.event.topology.LoadBalancer.DELETE
Message	DELETE loadBalancerId={0}.
Level	INFO
Explanation	loadBalancerId={0}のロードバランサーが削除されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.VIP.CREATE
Message	CREATE vipId={0}, data={1}.
Level	INFO
Explanation	vipId={0}のVIPが作成されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.VIP.UPDATE
Message	UPDATE vipId={0}, data={1}.
Level	INFO
Explanation	vipId={0}のVIPが{1}にアップデートされました。
Corrective Action	N/A

Logger	org.midonet.event.topology.VIP.DELETE
Message	DELETE vipId={0}.
Level	INFO
Explanation	vipId={0}のVIPが削除されました。
Corrective Action	N/A

プール

Logger	org.midonet.event.topology.Pool.CREATE
Message	CREATE poolId={0}, data={1}.
Level	INFO
Explanation	poolId={0}のプールが作成されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Pool.UPDATE
Message	UPDATE poolId={0}, data={1}.
Level	INFO
Explanation	poolId={0}のプールが{1}にアップデートされました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Pool.DELETE
Message	DELETE poolId={0}.
Level	INFO
Explanation	poolId={0}のプールが削除されました。
Corrective Action	N/A

プールメンバー

Logger	org.midonet.event.topology.PoolMember.CREATE
Message	CREATE poolMemberId={0}, data={1}.

Logger	org.midonet.event.topology.PoolMember.DELETE
Message	DELETE poolMemberId={0}.
Level	INFO
Explanation	poolMemberId={0}のプールメンバーが削除されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.HealthMonitor.DELETE
Message	DELETE healthMonitorId={0}.
Level	INFO
Explanation	healthMonitorId={0}のヘルスマニターが削除されました。
Corrective Action	N/A

Logger	org.midonet.event.api.Nsdb.CONNECT
Message	NSDB クラスタに接続しました。
Level	INFO
Explanation	API サーバーは NSDB クラスタに接続していました。
Corrective Action	N/A

Logger	org.midonet.event.api.Nsdb.DISCONNECT
Message	NSDB クラスターから切断しました。
Level	WARNING
Explanation	APIサーバーはNSDB クラスターより切断されています。
Corrective Action	このイベント後、接続が復元されていたならば修正措置は必要ありません。もし このイベントが続くようであれば、APIサーバーとNSDB クラスター間のネットワーク接続を確認してください。

Logger	org.midonet.event.api.Nsdb.CONN_EXPIRE
Message	CONN_EXPIRE NSDB クラスターへの接続は期限切れです。
Level	ERROR
Explanation	APIサーバーからNSDB クラスターへの接続は期限切れです。
Corrective Action	APIサーバーとNSDB クラスター間のネットワーク接続を確認して、NSDB クラスターに再接続するようにMidoNet APIサーバーを再起動してください。

MidoNet エージェントイベント

このセクションではMidoNet Agentイベントに関連するメッセージについて説明します。

NSDB

Logger	org.midonet.event.agent.Nsdb.CONNECT
Message	NSDB クラスターに接続しました。
Level	INFO
Explanation	MidoNet AgentがNSDB クラスターに接続しました。
Corrective Action	N/A

Logger	org.midonet.event.agent.Nsdb.DISCONNECT
Message	DISCONNECT NSDB クラスターから切断されました。
Level	WARNING
Explanation	MidoNet エージェントはNSDB クラスターより切断されました。
Corrective Action	このイベント後、接続が復元されていたならば修正措置は必要ありません。このイベントが続くようであれば、MidoNet エージェントとNSDB クラスター間のネットワーク接続を確認してください。

Logger	org.midonet.event.agent.Nsdb.CONN_EXPIRE
Message	CONN_EXPIRE NSDB クラスターへの接続は期限切れです。MidoNet Agentを閉じてください。
Level	ERROR
Explanation	MidoNet AgentからNSDB クラスターへの接続は期限切れです。MidoNet Agentを閉じてください。
Corrective Action	MidoNet エージェントノードとNSDB クラスター間のネットワーク接続を確認し、NSDB クラスターに再接続されるよう、ノード上のMidoNet エージェントサービスを再起動してください。

Interface

Logger	org.midonet.event.agent.Interface.DETECT
Message	NEW interface={0}

Level	INFO
Explanation	MidoNet エージェントは新しいインターフェース={0}を検出しました
Corrective Action	N/A

Logger	org.midonet.event.agent.Interface.UPDATE
Message	UPDATEインターフェース={0}はアップデートされました。
Level	INFO
Explanation	MidoNet エージェントはインターフェース={0}内にアップデートを検出しました。
Corrective Action	N/A

Logger	org.midonet.event.agent.Interface.DELETE
Message	DELETEインターフェース={0}は削除されました。
Level	INFO
Explanation	MidoNet Agentはインターフェース={0}が削除されていることを検出しました。
Corrective Action	N/A

Service

Logger	org.midonet.event.agent.Service.START
Message	STARTサービスが開始されました。
Level	INFO
Explanation	サービスが開始されました。
Corrective Action	N/A

Logger	org.midonet.event.agent.Service.EXIT
Message	EXITサービスが終了しました。
Level	WARNING
Explanation	サービスが終了しました。
Corrective Action	意図せずにこのイベントが起こった場合、MidoNet Agentサービスを再起動してください。このイベントが繰り返されるようであれば、ディベロッパーが更なる調査をするため、バグトラッカー内でチケットを申請してください。

パケットトレーシング

MidoNet Agent (Midolman) 内で、(ロギング経由で)パケットトレーシングの設定をするには、'mm-trace' コマンドを使うことができます。

A MidoNet エージェントは、受信パケットをマッチングする際に、設定されたログのレベルに関わらずエージェントのログファイルのシミュレーションに関する全てのログをとるフィルターを持つことができます。

全てのトレースメッセージはパケットを識別するための"cookie:"プレフィックスを持っており、そのプレフィックスはトレーシングメッセージではないものをフィルタアウトするためのグレップ表現として使われます。



重要

フィルターは永続的ではなく、エージェントがリブートされる度に失われます。

しかしながら、mm-traceはまったく同じシンタックスのフィルターを表示し、そのフィルターを再追加できるようにするので、運用者がコマンドを簡単に再実行することを可能にします。

Usage

全ての利用可能なオプションは'--help' オプションとともに表示されます。

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

Subcommand: add - add a packet tracing match
-d, --debug          デバッグレベルでのログ
--dst-port <arg>    TCP/UDP送信先ポートのマッチ
--ethertype <arg>   イーサタイプとのマッチ
--ip-dst <arg>       IP送信先アドレスとのマッチ
--ip-protocol <arg> IPプロトコルフィールドとのマッチ
--ip-src <arg>       IP送信元アドレスとのマッチ
-l, --limit <arg>    このトレースを使用不可にする前にパケット数をマッチ
--mac-dst <arg>      送信先MACアドレスとのマッチ
--mac-src <arg>      送信元MACアドレスとのマッチ
--src-port <arg>     TCP/UDP送信元ポートとのマッチ
-t, --trace          トレースレベルでのログ
--help              ヘルプメッセージの表示

Subcommand: remove - パケット トレーシングマッチの除去
-d, --debug          デバッグレベルでのログ
--dst-port <arg>    TCP/UDP送信先ポートのマッチ
--ethertype <arg>   イーサタイプとのマッチ
--ip-dst <arg>       IP送信先アドレスとのマッチ
--ip-protocol <arg> IPプロトコルフィールドとのマッチ
--ip-src <arg>       IP送信元アドレスとのマッチ
-l, --limit <arg>    このトレースを使用不可にする前にパケット数をマッチ
--mac-dst <arg>      送信先MACアドレスとのマッチ
--mac-src <arg>      送信元MACアドレスとのマッチ
--src-port <arg>     TCP/UDP送信元ポートとのマッチ
-t, --trace          トレースレベルでのログ
--help              ヘルプメッセージの表示

Subcommand: flush - トレーシングマッチのリストの消去
-D, --dead-only      期限切れのトレーサーのみのフラッシュ
--help              ヘルプメッセージの表示

Subcommand: list - 全てのアクティブなトレーシングマッチのリスト化
-L, --live-only      アクティブトレーサーのみのリスト化
--help              ヘルプメッセージの表示
```

Example

```
$ 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)
```

目次

MidoNetはバーチャルエクステンシブルLAN(VXLAN)テクノロジーをサポートしています。 VXLANとはなにか?

このタイプのカプセル化技術(Ethernet-in-IP)は、VLANs(802.1q)と比べて、あるいはスタックされたVLANs(Q-in-Q)と比べても、ソフトウェアが定義したネットワークにとってははるかに適した技術です。

VXLANが従来型のVLANと比べて持つもう1つの大きな強みが、その24ビットのVXLAN IDです。このIDがあるおかげで、VXLANは1600万以上の論理ネットワークまで機能を増やすことができます。これに比べてVLANですとその数が最大で4096です。

- VXLANはMidoNetの中でどのようにサポートされているのでしょうか？*

MidoNetは次のものを通してVXLANの実装を行なっています。

*VXLANゲートウェイを用意することにより、アンダーレイに物理的なL3ホストを整備しつつオーバーレイを橋渡しします。

- MidoNetの各ホスト間にVXLANトンネリングを設けている。

VXLAN ゲートウェイ

VXLAN ゲートウェイ (VXGW) は、仮想ブリッジを、L3 ネットワークおよび VXLAN が使用可能な物理的スイッチを通じて連絡可能な物理的な L2 セグメントにまで延長することを可能にしてくれます。

VXLANが使用可能な物理的スイッチは、“ハードウェアVTEP”（VXLANトンネルエンドポイント）とも呼ばれている。VXGWは、1つのあるいはたくさんのVXLANベースのロジカルスイッチを作成することを許可し、これらのスイッチは好きなだけの数のハードウェアVTEPに広がることができ、また単一のMidoNet-ODPクラウドにも広がります。

VXGWには次のような利点があります。

**物理的なL2セグメントの中で、オーバーレイやサーバー内において、VM間にL2の接続性を提供します。

MidoNetコントローラーは、学習したMACを、全てのVTEP間およびMidoNetのネットワークステートデータベース(NSDB)間で自動交換します。

コーディネーターは、MidoNetアーキテクチャーの構成要素であり、VXLANサポートを担当しています。

*MidoNet REST APIを通じて、VTEPの状態を開示します。

- VTEPスイッチを設定することによって、MidoNet REST APIを通じて設定したバインディングを実装します。
- MNとVTEPとの間に流れるトラフィックにたいして、L2制御プレーインの役割を果たします。

MidoNetをハードウェアVTEPに接続する時にはこの手順を踏みます。あるVTEP上で、いずれかのニュートロネットワークをポート/vlanペアにバインドしようとする場合には、その前に、必ずこの手順を踏む必要があります。

1. 手元にあるスイッチの文書を参照して、スイッチ上でVXLANを有効化し、スイッチに必要なパラメータ全てを備えたVTEPとして設定します。

MidoNetは、VTEP上のPhysical Switchテーブルには、このVTEPのマネージメントIP、マネージメントポートおよびトンネルIPを含むレコードが入っているものと予想します。これら詳細情報は手元に置いておきましょう。これらの詳細情報はVTEPを設定する時、あるいはニュートロンネットワークへのなんらかのバインディングを設定する時には必要になるからです。このテーブルのコンテンツを別場所へ書きだす(ダンプする)には次のコマンドを使います。

取り扱っているVTEPが、全ての物理的なポートを確実に登録するよう注意を払います。VTEPの中にあるPhysical_Portsテーブルを見れば、登録を検証することができます。このテーブルが表示するポートのみがニュートロンネットワークにバインドさせるものとして利用できます。次のコマンドを使えば物理的なポート全てが表示されますが、その時、Physical_Switchに付与した名前が<vtep_name>に取って代わります。（この点は、前記したコマンド”vtep-ctl list Physical_Switch”を使うことで確認することができます。）

2. VTEPを設定した後、トンネルとの接続状態ならびにマネジメントインターフェースとの接続状態の両方をテストする必要があるかもしれません。いずれの接続状態とも、'up' 状態であるべきです。

マネジメントデータベースへの接続状態をテストするには次の内容を実行します。

```
$ telnet <management-ip> <management-port>
Trying <management-ip>...
```

```
Connected to <management-ip>  
Escape character is '^['.
```

この時点で、これをコンソールにペーストします。

```
{"method": "list_dbs", "id": "list_dbs", "params": []}
```

そうすると、この返信内容を確認することができます。

```
{"id":"list_dbs","error":null,"result":["hardware_vtep"]}
```

このVTEPの設定を保持しているOVSDDBサーバーが、稼働中であり接続を担当しているのだということを検証したところです。同じような返信を受信しなかった場合には、VTEPの設定をもう一度調べてください。

3. `midonet-api config file/usr/share/midonet-api/WEB-INF/web.xml` を修正して、Midonetの中のVXLANサービスを有効にします。
 - a. `midonet-api config file/usr/share/midonet-api/WEB-INF/web.xml`の中で、このsnippetを見つけます。

```
<!-- VXLAN gateway configuration -->
<context-param>
  <param-name>midobrain-vxgw_enabled</param-name>
  <param-value>>false</param-value>
</context-param>
```

そして、`<param-value>`タグのバリューを' true' に変更してください。

- b. 変更を適用するためにTomcatを再起動します。
4. VTEPが正しく設定できたことを確認できれば、VTEPをMidoNet設定に追加することができます。

詳細につきましては、「[VTEPの追加](#)」 [\[100\]](#)をご覧ください。



重要

VLAN-ポートの割り当て情報、VTEPマネジメントインターフェースIP およびそのポートに関する情報の他にも、”VTEP”種別のTunnelZoneの識別子情報が必要です。MidoNet Agentデーモンを実行しているホストのうち、VTEPを使ってVXLANとのVXLANトンネルを作成したいホストについては、このトンネルゾーンのメンバーにならなければなりません。この時には、個々のホストがVXLANトンネルのエンドポイントとして使っているローカルIPを使います。

MidoNet設定にVTEPを追加できましたら、APIサーバーはその(VTEPの)マネジメントインターフェースに接続し、ロジカルブリッジを作成するために必要な情報の全てを収集します。さらに詳しいことにつきましては”Logical Bridge”の章をご覧ください。

VTEPとニュートロンネットワークの接続設定

MidoNetの中で、VTEPとニュートロンネットワークとの間の接続を設定する時にはこの手順を踏みます。

本手順を遂行するには、次に挙げる情報を把握しておく必要があります。VTEPのマネージメントIPならびにそのポート、VTEP上にある物理的なポートおよび接続しようとしているこのポートのVLAN ID、VTEPに接続させようとしているニュートロンネッ

$+$

```
midonet> tunnel-zone tzone1 add member host host0 address 172.16.0.1
```

ニュートロネットワークの中にあるホストの中でVTEPと通信させたいホスト全てについて、その1つずつにたいしてこの手順を繰り返し踏みます。



重要

通常、ホストはgreタイプかvxlanタイプかのいずれか1つのトンネルゾーンにしか割り当てられません。VTEPに接続しているホストというのは例外なのです。なぜならば、その場合には2つのトンネルゾーンに、つまりgre/vxlanタイプとvtepタイプの両方に割り当てることができるからです。そしてこの場合でも、ホストは両方のトンネルゾーンにたいして同じIPを使用することができることを覚えておきましょう。

VTEPのvlan10とニュートロンネットワークとの間のバインディングをbridge0の背後に作成します。・+ この事例では、bridge0の背後にvlan10上のホスト各種をニュートロンネットワーク765cf657-3cf4-4d79-9621-7d71af38a298と接続しています。

+

```
midonet> vtep management-ip 192.168.2.11 binding add network-id
765cf657-3cf4-4d79-9621-7d71af38a298 physical-port swp1s2 vlan 10
```

MidoNetの中で、VTEPのvlan10物理ポートswp1s2の背後にあるネットワークと、UUIDが765cf657-3cf4-4d79-9621-7d71af38a298のニュートロンネットワークとの間にバインディングを作成に成功しました。



ヒント

VTEPとMidoNetとの間の接続をテストする（つまりVTEP上のホストからMidoNetを”ping”すること）ためには、MidoNetにホストのIPアドレスを追加することによって、進入セキュリティルールを修正する必要があります。（MidoNetからホストをpingするために別途なicaを追加で設定する必要はありません。）さらに詳しいことにつきましては[ref:connect vtep to midonet\[\]](#)を参照してください。

VTEPとMidoNetホストの接続

ニュートロンの初期設定は、そのネットワーク全てについてのセキュリティールールを含んでいるため、そのネットワーク上のVMのIP/MACが宛先であるトラフィックにしか送信がされないよう制限がかかっています。

VTEPの中であるネットワークを物理的なポートにバインドすることで、事実上、ニュートロン自体が認識していないこのニュートロンネットワークのL2セグメントに、ホストを追加していることになるのです。したがって、これらの物理的なホストに向けたトラフィックはドロップされることになります。

以下に挙げた手順は、VTEP上のホストへのトラフィックを許可するために、初期設定されている進入セキュリティルールをどのように変更するかを示しています。

1. このコマンドを発行し、初期設定されている進入セキュリティールールがいかなるものなのかをMidoNet CLIの中で確認します。

```
midonet> list chain
chain chain0 name OS SG 64d9f3df-9875-4896-ad0c-ffc5bba84c5e INGRESS
```


もしもMidoNetが修復不可能なエラーをみつけた場合には、次のWARNメッセージが表示されます(マネージメントポートおよびidは前記のものと同じであることが前提)

192.168.2.13 : 6632において、VTEPを上手くブートストラップすることができませんでした。

MidoNetコーディネーターは、このニュートロンネットワークが再びアップデートされるまではこのニュートロンネットワークを無視します。MidoNetコーディネーターは、設定されているその他のネットワークとの動作は継続することができます。

- MidoNet コーディネーターが状況と同期を取っていることを確認します。*

この時点までエラー表示が全くなかった場合には、上述の logback.xml ファイルを編集し、vxgwプロセスの中でDEBUGログを有効にします。

```
<!-- <logger name="org.midonet.vxgw" level="DEBUG" /> -->
```

❯!- and -> タグを取り除くことでこの設定を有効にして、APIログがDEBUGメッセージを表示し始めるまで数秒間待ちます。さらに細かい情報を見るにはDEBUGではなくTRACEを選択します。(パフォーマンスに大きく影響を与えてしまうほどに冗長な情報はありません。)

以下のようなメッセージは、MidoNetコーディネーターがMidoNetとVTEP間でMACどうしを交換することに成功したことを示しています。

```
TRACE c68fa502-62e5-4b33-9f2f-d5d0257deb4f - Learned: MacLocation { logicalSwitchName=mn-  
c68fa502-62e5-4b33-9f2f-d5d0257deb4f, mac=96:8f:e8:12:33:55, vxlanTunnelEndpoint=192.168.  
2.16 }
```

このメッセージは、与えられているMACに関するアップデートが、ニュートロンネットワークc68fa502-62e5-4b33-9f2f-d5d0257deb4fに所属するロジカルスイッチ上でみつかったことを示しています。この場合、vxlanトンネルエンドポイントは192.168.2.16だったということで、つまりMACはトンネルエンドポイントで見つけることができることを示しています。ポートからMACを取り除かれたことが、vxlanTunnelEndpoint=null(これは「MACはいずれのポートにもいません」という意味)という文字で判ります。

VxLANトンネルが確立したことを検証します

コーディネーターが正常に作動しているにもかかわらず、トラフィックがいまだに流れないのであれば、VTEPsならびにMidoNetホストが上手くVxLANトンネルを確立できるか検証すべきです。

VMから通信したい相手先サーバーへのpingを稼働させながら、VTEP上の相手先サーバーとの通信を試みているVMのホストであるMidoNet コンピュートにログインします。次のコマンドを実行します。

```
tcpdump -leni any port 4789
```

MidoNetコンピュートが192.168.2.14であることを前提とし、また、VTEPのトンネルIPが192.168.2.17であることも前提にすると、出力内容は以下のような内容となります（お使いのtcpdumpバージョンに応じて変わります。）

```

15:51:28.183233 Out fa:16:3e:df:b7:53 ethertype IPv4 (0x0800), length 94: 192.168.2.14.
39547 > 192.168.2.17.4789: VXLAN, flags [I] (0x08), vni 10012
aa:aa:aa:aa:aa:aa > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806), length 42: Request who-has
10.0.0.1 tell 10.0.0.10, length 28
15:51:28.186891 In fa:16:3e:52:d8:f3 ethertype IPv4 (0x0800), length 94: 192.168.2.17.
59630 > 192.168.2.13.4789: VXLAN, flags [I] (0x08), vni 10012
cc:dd:ee:ee:ee:ee > aa:aa:aa:aa:aa:aa, ethertype ARP (0x0806), length 42: Reply 10.0.0.10
is-at cc:dd:ee:ee:ee:ff

```


次のような条件の中ではコマンドの遂行は失敗に終わります。

- もしもそのポート-VLANペアが、既にもう1つ別のニュートロンネットワークに橋渡しされていた場合
- そのニュートロンネットワークが、既に、もう1つ別のハードウェアVTEP上にあるポート-VLANペアに橋渡しされていた場合

事例

成功したコマンド

```
midonet> vtep management-ip 119.15.112.22 add binding physical-port in1 vlan 143 network-
id 9082e813-38f1-4795-8844-8fc35ec0b19b
management-ip 119.15.112.22 physical-port in1 vlan 143 network-id
9082e813-38f1-4795-8844-8fc35ec0b19b
```

成功しなかったコマンド

```
midonet> vtep management-ip 119.15.112.22 add binding physical-port in1 vlan 143 network-
id 9082e813-38f1-4795-8844-8fc35ec0b19b
Internal error: {"message": "内部サーバーエラーが発生しましたので、再度トライしてみてください。", "code": 500}
midonet> vtep management-ip 119.15.112.22 add binding physical-port in1 vlan 144 network-
id 9082e813-38f1-4795-8844-8fc35ec00000
Internal error: {"message": "No bridge with ID 9082e813-38f1-4795-8844-8fc35ec00000 exists.", "code": 400}
```

VTEPバインディング

MidoNet CLIは、与えられているVTEP上の全てのバインディングについての説明を入手するためのコマンドを提供しています。また、MidoNet CLIは、特定のニュートロンネットワークがバインドしているVTEP全てについての説明を入手するためのコマンドも提供しています。

- VTEPの中にある全てのバインディング*

はじめに、VTEP全てをリスト化して、適切なマネージメントIPが特定できるようにします。

```
midonet> vtep list
name vtep0 description Vtep1 management-ip 192.168.2.13 management-port 6632 tunnel-zone
tzone0 connection-state CONNECTED
```

結果

コマンドが成功しますと、プログラムは、選択したVTEP上にある全てのVXLANポートに関する説明およびそれらVXLANポートとニュートロンネットワークとのバインディング情報を返信します。

```
vtep management-ip 192.168.2.13 list binding
binding binding0 management-ip 192.168.2.13 physical-port Te 0/2 vlan 908 network-id
bc3afc36-6274-4603-9109-c29f1c12ba33
binding binding1 management-ip 192.168.2.13 physical-port Te 0/2 vlan 439 network-id
1d475afc-d892-4dc7-af72-9bd88e565dde
binding binding4 management-ip 192.168.2.13 physical-port in1 vlan 119 network-id
bc3afc36-6274-4603-9109-c29f1c12ba33
```

出力した内容結果を見ると、与えられたVTEPに適用したポート-vlanペア全てをリストで見ることができます。以下の情報が表示されています(一行目は事例として使用しています)。* バインディングのエリア (たとえば binding0)。


```
midonet> vtep management-ip 119.15.112.22 delete binding
Syntax error at: ...binding
midonet> vtep management-ip 119.15.112.22 delete binding physical-port in1
Syntax error at: ...binding physical-port in1
```

VTEPの削除

このコマンドはVTEPを削除する時に使用してください。

シンタックス

```
vtep management-ip vtep-ip-address delete
```

結果

このコマンドを発行すると、MidoNetがリスト化しており認知されているVTEPからVTEPを完全に削除します。

このコマンドは、VTEPのポートVLANペアのいずれかがニュートロンネットワークのいずれかにバインドしている場合は成功しません。

事例

```
midonet> vtep management-ip 119.15.120.123 delete
```



注記

別の方法としては、VTEPのポートVLANペア全てをニュートロンネットワークから切り離すためには、vxlanポートを削除するという方法があります。

目次

本セクションでは、MidoNetのバーチャルブリッジとフィジカルスイッチ間のL2ゲートウェイのセットアップ方法について記載しています。



下記の図が典型的なL2ゲートウェイトポロジーを表しています。

The diagram illustrates a VAB (Virtual Access Bridge) setup. At the top, two physical switches, each labeled "Phys. Switch" and featuring a bidirectional arrow icon, are connected to a central VAB via "Trunk" links. The central VAB is represented by a brown shape with a curved top and is labeled "VAB" in the center. It is divided into two sections: "VUB-8" on the left and "VUB-5" on the right. Below VUB-8, two "VM" (Virtual Machine) icons are connected. Similarly, below VUB-5, two "VM" icons are connected.


```
bridge0:port3
```

4. "host0:eth0" と "host1:eth1" の二つのインターフェイスがフィジカルスイッチのトランクに接続されていると仮定し、それらのインターフェイスをVABのトランスポートに紐づけます。

```
midonet> host host0 binding add interface eth0 port bridge0:port2
midonet> host host1 binding add interface eth1 port bridge0:port3
```

フェイルオーバーとフェイルバック

フィジカルブリッジ上で可能になったスパンニングツリープロトコル (STP) とのコンビネーションにより、MidoNet VABはフェールオーバー機能を提供しています。これは、トランクポートを超えたブリッジプロトコルデータユニット (BPDU) フレームを転送することで可能になります。

STPがあるため、両方のフィジカルスイッチが同じブリッジネットワークに属すると仮定し、デバイスがMidoNet VABを通過するループを探知します。そして、ひとつのスイッチはトランクをブロックするよう選択されます。例として、レフトスイッチがブロックされると仮定してみます。VABはライトトランクより入ってくるトラフィックのみをみます。従って、フレーム内にみられる全てのMACアドレスのソースをライトトランクへと関連付けます。

ネットワーク障害を含む様々なイベントは、トランクのステートの変換をもたらす可能性があります。例としては、MidoNetがレフトスイッチとの接続を失った時に、BPDUがライトブリッジへ（あるいは、ライトブリッジから）転送されなくなり、ループが終わってしまうことが挙げられます。

そのようなフェイルオーバーのシナリオでは、他のスイッチからトラフィックが流れることになります。今回の更新により、MidoNetは新たなポートでのインカミングトラフィックを検知し、内部のMACポートとの結合をアップデートします。トポロジーの以前のステートが復元されたとしても（つまり、MidoNetがレフトスイッチとの接続を復旧することを意味します）、MidoNetはそれを探知して、MACポートとの結合をアップデートします。

フェイルオーバーやフェイルバックにかかる時間は主に”フォワードディレイ”、スイッチ上のSTP設定やトラフィックの性質によります。トランクより継続的なインカミングトラフィックがある場合、標準値としては、MACが学習する時間を合わせてフェイルオーバーやフェイルバックのサイクルは50秒で完了します。

目次

CLIを通すことでMidoNetのバーチャルトポロジを全て調べて編集することができますが、実際に実行する際は細心の注意を払って行います。なぜならば、MidoNetのバーチャルネットワークに対する考え方とOpenStackの見え方の間に矛盾が生じやすいからです。



矛盾が生じないように意識しつつ、CLIを役立てるためのいくつかのタスクを見ていきます。

- ## MidoNet CLIの使用

1. SSHを使用してMidoNet CLIが作動しているホストと接続します。

接続されているデバイスのIPアドレス、ユーザーネームやパスワードなどのログイン情報が必要となります。SSHコマンドの例は以下のようなソースが挙げられます。

ユーザーネームは既に入力済みなので、'root' というコマンドを入力します。
サーバーがパスワードを要求するので、あなたのパスワードを入力します。

2. CLIはマニュアルページに記載されています。マニュアルページを閲覧するには以下のソース例にあるコマンドを入力します。

111

```
$ mn-conf dump -s
```

このセクションには、MidoNetのパフォーマンスに影響を及ぼす推奨設定の情報が含まれています。

mn-conf(1) の agent.midolman セクション

simulation threads - パケット処理に専属的に使われるスレッドの数

output_channels - フロー作成とパケット実行に活用されるアウトプットチャネルの数。各チャネルには専属スレッドがあります。

`input_channel_threading` - 各データポートからパケットを受け取る為に、MidoNet エージェントは、専属のNetlinkチャンネルを作成します。このオプションは、それらのチャンネルにおける、読み込みのスレッドストラテジーをチューニングします。`one_to_many`の場合は、全てのインプットチャンネルに対して、エージェントは一つのスレッドを使います。`one_to_one`の場合は、各インプットチャンネルに対して、エージェントが一つのスレッドを使います。

mn-conf(1) の agent.datapath セクション

global_incoming_burst_capacity - パケットが処理されてシステムをだて行く時にリフィルされるHierarchical Token Bucket (HTB)によって、入ってくるパケットはレート制限されます。この設定は、HTTBの中のルートバケットのパケット内のサイズをコントロールします。デーモンがパケットのドロップを始める前に、バースト内（ハンドルできる高いほうのレート）で受け入れるパケットの数になります。それは、システム内のインフライトパケットの最大数です。この値の変更は、ハイスループットのレイテンシーに影響します。

`tunnel_incoming_burst_capacity` - トンネルポートはHTTBで自らのバケット取ることができます。最大レートで送られない時に、トンネルポートがバーストキャパシティ蓄積することを許可します。この設定は、そのバケットのパケット内のサイズをコントロールします。

`vm_incoming_burst_capacity` - 各VMポートは、HTBの中で自らのバケットを獲得します。最大レートで送られない時に、トンネルポートがバーストキャパシティを蓄積することを許可します。この設定は、そのバケットのパケット内のサイズをコントロールします。

mn-conf(1) の agent.loggers セクション

root

loglevel - デフォルトのログレベルです。DEBUG設定は、開発とトラブルシューティングのみに利用します。この設定はパフォーマンスに影響する上、非常に冗長的なためです。

midolman-env.sh

MAX HEAP SIZE - JVMに割り当てられるメモリの総量

HEAP_NEWSIZE - エデンのガベージコレクション生成に使われるJVMメモリーの総量です。パケット処理の際に短命なオブジェクト向けにエージェントが使うメモリーの量は、概算して全体の75%です。

推獎值

表17.1 推奨される設定値

File	Section	Option	Compute	Gateway	L4 Gateway
logback.xml	root	level	INFO	INFO	INFO
midolman-env.sh	-	MAX_HEAP_SIZE	2048M	6144M	6144M
midolman-env.sh	-	HEAP_NEWSIZE	1536M	5120M	5120M
mn-conf(1)	[agent.midolman]	simulation_threads	4	4	16
mn-conf(1)	[agent.midolman]	output_channels	1	2	2
mn-conf(1)	[agent.midolman]	input_channel_threading	one_to_many	one_to_many	one_to_many
mn-conf(1)	[agent.datapath]	global_incoming_buffer_size	256	256	128
mn-conf(1)	[agent.datapath]	tunnel_incoming_buffer_size	128	128	64
mn-conf(1)	[agent.datapath]	vm_incoming_burst_capacity	16	32	16

MidoNet エージェント (Midolman) 設定オプション

このセクションは、MidoNet エージェントの設定オプションすべてをカバーします。

zookeeper.session.gracetime と agent.datapath.send_buffer_pool_buf_size_kb の設定値のみを除いて、デフォルトバリューの変更は推奨しません。



警告

本当に必要が無い限り、ルートキー、クラスター名、キースペースの変更を避けてください。

114

```
zookeeper_hosts = <カンマ区切りのIPアドレス>
session_timeout = 30000
root_key = /midonet/v1
session_gracetime = 30000
}
```

Cassandra 設定

以下を調整する為にCassandra設定を活用できます。

- データベースの複製ファクター
- MidoNetクラスター名

```
cassandra {
  servers = <カンマ区切りのIPアドレス>
  replication_factor = 1
  cluster = midonet
}
```

データパス設定

エージェントは、データパスにリクエストを送信するために再利用可能なバッファのプールを使用しています。プールサイズとバッファを調整するために、mn-conf(1)のagent.datapathのオプションを使用することが可能です。各出力チャンネルにひとつのプールが作成され、それぞれに適用されます。

パケットサイズが、最大のバッファサイズを超えてしまったために、パフォーマンスが落ちてしまったことに気づいたときは、buf_size_kb設定の値を上げることができます。この設定はバッファサイズ (KB単位) をコントロールします。このバッファサイズはMidoNetエージェントが送ることができるパケットサイズの上限を規定します。Jumboフレームが横切るネットワークの中では、サイズを調整しましょう。そうすることで、一つのバッファが全体のフレームに乗っかることができ、フローアクションのために十分な余力も残すことができます。

BGP フェールオーバー設定

デフォルトのBGPフェールオーバー時間は2,3分です。しかし、セッションの両端のいくつかのパラメーターを変えることによってこの時間を減らすことができます: mn-conf(1) (MidoNet側) とBGPピア設定のリモート側です。下記の事例は、MidoNet側でフェールオーバー時間を1分に減らすやり方を示している事例です。

```
agent {
  midolman {
    bgp_connect_retry=1
    bgp_holdtime=3
    bgp_keeplive=1
  }
}
```

ホストのmn-confの設定は、BGPピア設定のリモートエンドのものとマッチしている必要があります。設定に関するより詳細な情報は「[BGPピアにおけるBGPフェールオーバーの設定](#)」[4]をご参照ください。

より高度なMidoNet API設定オプション

このセクションはより高度なユーザーが活用できる設定要素について記しています。より高度なMidoNet設定運用を実行するため設定要素を使うことができます。MidoNet API設定は /usr/share/midonet-api/WEB-INF/web.xml ファイルに保存できます。

以下は、web.xmlスニペットの事例です。



注記

残りの設定は、クラウドコントローラーに依拠しており、このドキュメントの関連するセクションでカバーされています。

zookeeper-zookeeper_hosts

ZooKeeperホストのリストはMidoNet設定データを格納するために使われます。エンタリーはコンマ区切りになります。

```
<context-param>
  <param-name>zookeeper-zookeeper_hosts</param-name>
  <param-value>192.168.1.100:2181,192.168.1.101:2181,192.168.1.102:2181</param-value>
</context-param>
```

zookeeper-session_timeout

ZookeeperがZookeeperサーバーからクライアントをディスコネクトすることを検討してから、タイムアウトバリュー（ミリ秒単位）で設定します。

```
<context-param>
  <param-name>zookeeper-session_timeout</param-name>
  <param-value>30000</param-value>
</context-param>
```

Tomcatの始動時間の改善

ある特定の状況においては、サービスが始まる前に、Tomcatを使うことがあります。場合によっては最大10分使うことがあります。

Midonet APIをTomcat 7が乗っかっているUbuntu 14.04にデプロイする時に使う可能性が非常に高くなっています。Tomcatの始動時間を改善する為の良い方法は、/usr/share/tomcat7/bin/catalina.shファイルに以下を設定することです。

```
JAVA_OPTS="$JAVA_OPTS -Djava.security.egd=file:/dev/./urandom"
```

詳細に関しては、こちらを参照ください。 <http://wiki.apache.org/tomcat/HowTo/FasterStartup>.

第18章 MidoNet と OpenStack TCP/UDP サービスポート

目次

コントローラーノードのサービス	120
ネットワークステータデータベースノードサービス	121
コンピュートノードのサービス	122
ゲートウェイノードサービス	122

このセクションはMidoNet とOpenStackのサービスで使うTCP/UDPポートをリスト化します。

コントローラーノードのサービス

このセクションはコントローラーノサービスを使われるTCP/UDPポートのリスト化をしています。

Category	Service	Protocol	Port	Self	Controller	Compute	Mgmt. PC
OpenStack	glance-api	TCP	9292	x	x	x	x
OpenStack	httpd (Horizon)	TCP	80	x			x
MidoNet	midonet-api	TCP	8080	x	x		x
OpenStack	swift-object-server	TCP	6000	x	x	x	
OpenStack	swift-container-server	TCP	6001	x	x	x	
OpenStack	swift-account-server	TCP	6002	x	x	x	
OpenStack	keystone	TCP	35357	x	x	x	x
OpenStack	neutron-server	TCP	9696	x	x	x	x
OpenStack	nova-novnc-proxy	TCP	6080	x	x		x
OpenStack	heat-api	TCP	8004	x	x		x
OpenStack	nova-api	TCP	8773	x	x		x
Tomcat	Tomcat shutdown control channel	TCP	8005	x	x		
OpenStack	nova-api	TCP	8774	x	x	x	x
OpenStack	nova-api	TCP	8775	x	x	x	x
OpenStack	glance-registry	TCP	9191	x	x	x	
OpenStack	qpid	TCP	5672	x	x	x	
OpenStack	keystone	TCP	5000	x	x	x	x
OpenStack	cinder-api	TCP	8776	x	x	x	x

ネットワークステートデータベースノードサービス

Category	Service	Protocol	Port	Self	Controller	NSDB	Compute	Comment
MidoNet	ZooKeeper communication	TCP	3888	x		x		
MidoNet	ZooKeeper leader	TCP	2888	x		x		
MidoNet	ZooKeeper/Cassandra	TCP	random	x				ZooKeeper/Cassandra はTCPハイナンバーポートを“LISTEN”します。各 ZooKeeper/Cassandra ホストでポート番号はランダムに選択されます。
MidoNet	Cassandra Query Language (CQL) native transport port	TCP	9042					
MidoNet	Cassandra cluster communication	TCP	7000	x		x		
MidoNet	Cassandra cluster communication (Transport Layer Security (TLS) support)	TCP	7001	x		x		
MidoNet	Cassandra JMX	TCP	7199	x				もし Cassandra の健全性をモニターする為にこのポートを使っているなら、JMX モニター

Category	Service	Protocol	Port	Self	Controller	NSDB	Compute	Comment
								ポートはモニターサーバーからのコミュニケーションを可能にします。
MidoNet	ZooKeeper client	TCP	2181	x	x	x	x	
MidoNet	Cassandra clients	TCP	9160	x	x	x	x	

コンピュートノードのサービス

このセクションはコンピュートノードのサービスで使われるTCP/UDPポートのリスト化をしています。

Category	Service	Protocol	Port	Self	Controller	Comment
OpenStack	qemu-kvm vnc	TCP	From 5900 to 5900 + # of VM		x	
MidoNet	midolman	TCP	random	x		midolemanはTCPハイナバーポートを“LISTEN”します。各MNエージェントホストでポート番号はランダムに選択されます。
OpenStack	libvirtd	TCP	16509	x	x	
MidoNet	midolman	TCP	7200	x		もし健全性をモニターする為にこのポートを使っているなら、JMXモニターポートはモニターサーバーからのコミュニケーションを可能にします。

ゲートウェイノードサービス

このセクションはゲートウェイノードのサービスによって使われるTCP/UDPポートをリスト化しています。

Category	Service	Protocol	Port	Self	Misc.	Comment
MidoNet	midolman	TCP	random	x		midolemanはTCPハイナバーポートを“LISTEN”します。各MNエージェントホストでポート番号はランダムに選択されます。

T - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT - DRAFT