# 活用研究会を通じた NICT総合テストベッド 利用体験事例

東京大学 生産技術研究所
山本成一
yama@iis.u-tokyo.ac.jp

# 自己紹介

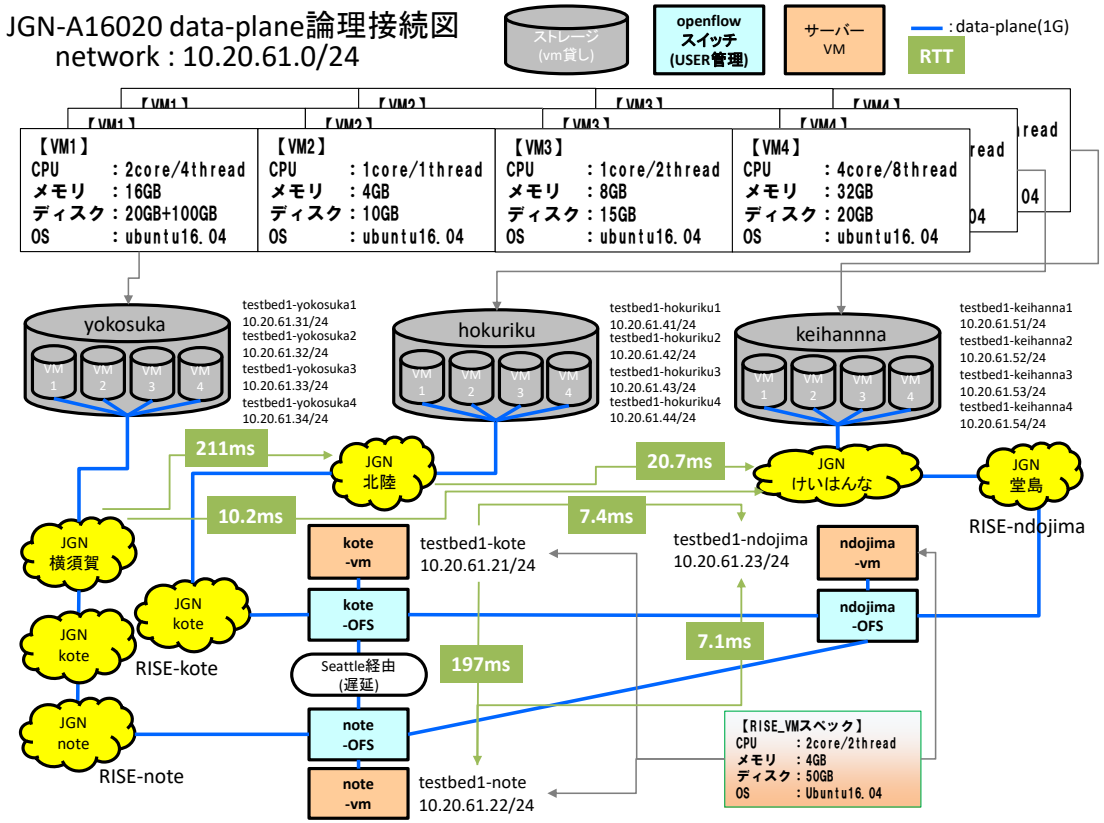
- 東京大学 生産技術研究所 電子計算機室 助教
  - 研究所のネットワークおよびコンピュータシステムの設計、運用
  - トラブル対応、利用者対応、等
- 情報通信研究機構 招へい専門員
  - 総合テストベッド研究開発推進センター
    - おもにJGNの運用関係での助言等
    - JGN網設計、サービス考案、スキーム作りに関わる
- その他(JGN関連)
  - 東大平木研実験参加
    - Internet2 Land speed record等
  - WIDEプロジェクト
    - JBプロジェクト/プログラマブルIXに関する研究開発、等
- (利用者としての)テストベッドの利用経験
  - あり
    - JGNおよび(初期の)StarBED
  - 「活用研究会」での利用は「なし」 => 今回は利用者視点で体験してみる

# 目標

- 「活用研究会を通じたNICT総合テストベッド利用体験事例」
  - 実験利用を想定した一例を考案
  - 想定例にて動作確認を行う
  - 利用時に気がついた点をメモ

- テストベッドの利用者層、利用形態は多種多様
  - あくまで「ネットワークを専門とした一研究者の視点」にて

# 「活用研究会」の資源




- JOSE環境
  - VM 12台
    - OS: Ubnutu Linux 16.04
    - NIC: x2
      - 実験用と管理用の2種
      - いずれもIPv4 /24の割当
    - 管理者権限(root)あり
- RISE環境
  - Openflow sw x3
  - VM x3
- 注意
  - インターネット接続性は許可制
  - 踏み台を利用してsshログイン
  - 「共用」のため、他利用者へ影響しないように

本発表は VM12台 のみを利用

# 実験利用の想定

- (例)「(IP)ネットワーク」に関する研究
  - 任意の(論理)トポロジを利用
  - L2/L3パスの利用
  - 遅延の挿入
  - 数台-数百台のルータ・ホストの利用

# 実験利用の想定

- 「(IP)ネットワーク」に関する研究
  - 任意の(論理)トポロジを利用
  - L2/L3パスの利用
  - 遅延の挿入
  - 数台-数百台のルータ・ホストの利用

＋ **「共用」の制限**

「仮想化(論理化)技術」の利用にて対応

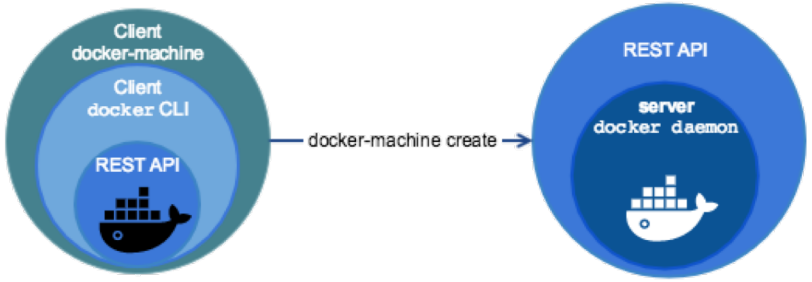
# 仮想化技術の選定

- ホスト: Ubuntu Linux 16.04
- 選択肢(主観的コメントを含む)
  - Linux KVM/Xen
    - 任意のOSを隔離して動作させる機構。単一ホスト内で動作。Openstack等の統合環境操作ツールはあるが、そもそも入れ子(nested)動作のためCPUオーバヘッドが大きい。
  - netns
    - 名前空間(ネットワーク、プロセス、等)の分離機能。直接操作は運用コストが大きい。
  - mininet
    - netnsの応用。主に単一ホスト内で動作。主に学術利用。
  - docker
    - netnsの応用。アプリケーション仮想化を目的に作成された。
      商用でも多く利用。
      **ツールやイメージ豊富。ポータビリティに富み、複数ホストでの連携動作可。**
  - LXD
    - netnsの応用。DockerよりOS仮想化を目的に作成された。

本発表では
dockerを選定

# dockerの利用

- docker machine
  - 複数ホストで動作するdockerをまとめて操作
  - ネットワーク管理機能もあり
    - overlayドライバにてホスト間での同一セグメント利用可能
- 利用イメージ(アプリケーション)
  - vyos
    - IPルーティング機構の統合パッケージ
    - Zebra => quagga => (vyatta) => vyos (最近はfrrもあり)
    - docker環境下では、特権(priviledge)モードによりIP転送可





https://docs.docker.com/machine/overview/

https://vyos.io/

http://www.vyos-users.jp/

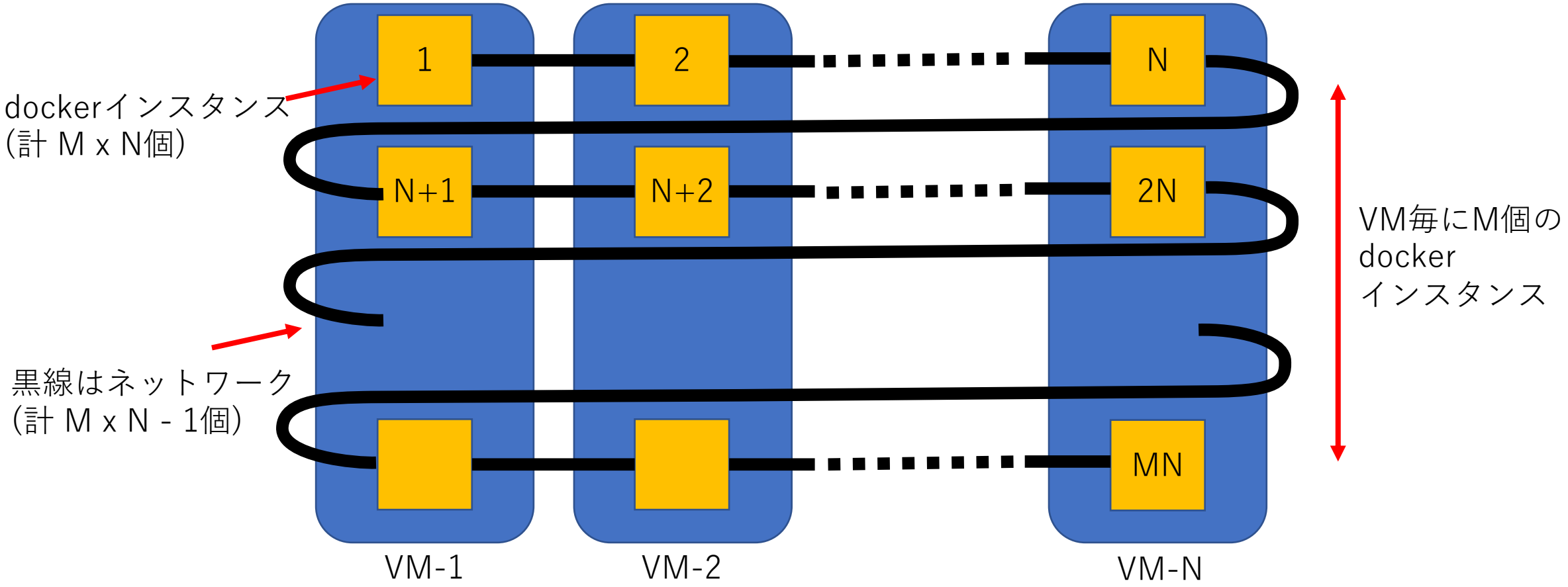
# docker環境構築(概要)

- 全台(x12)
  - インターネット接続設定(gateway, DNS)
  - ssh鍵配置
- 操作用VM x1
  - GNU screenの利用
    - セッションが切れても処理を継続できる
    - ラッパのbyobuを利用
  - docker-machineのインストール
    - Ubuntu標準のdocker.ioパッケージから、docker-ceパッケージへ入替え
- docker node x11
  - 全台にて、docker-machineからの遠隔操作許可設定(sudo許可)
  - DB用VM設定 x1 (swarm-kvs)
    - DB用dockerイメージ(consul)の展開
  - 実験インスタンス動作用VM設定 x10 (swarm-nodeX)
    - マスタ x1 (swarm master)
    - スレーブ x9

# docker環境構築
# 構築したdocker node一覧

```
yama@testbed1-yokosuka1:~$ docker-machine ls
NAME           ACTIVE      DRIVER    STATE     URL                         SWARM                      DOCKER        ERRORS
swarm-kvs      -           generic   Running   tcp://192.168.61.32:2376                               v18.06.1-ce
swarm-node0    * (swarm)   generic   Running   tcp://192.168.61.33:2376    swarm-node0 (master)       v18.06.1-ce
swarm-node1    -           generic   Running   tcp://192.168.61.34:2376    swarm-node0                v18.06.1-ce
swarm-node2    -           generic   Running   tcp://192.168.61.41:2376    swarm-node0                v18.06.1-ce
swarm-node3    -           generic   Running   tcp://192.168.61.42:2376    swarm-node0                v18.06.1-ce
swarm-node4    -           generic   Running   tcp://192.168.61.43:2376    swarm-node0                v18.06.1-ce
swarm-node5    -           generic   Running   tcp://192.168.61.44:2376    swarm-node0                v18.06.1-ce
swarm-node6    -           generic   Running   tcp://192.168.61.51:2376    swarm-node0                v18.06.1-ce
swarm-node7    -           generic   Running   tcp://192.168.61.52:2376    swarm-node0                v18.06.1-ce
swarm-node8    -           generic   Running   tcp://192.168.61.53:2376    swarm-node0                v18.06.1-ce
swarm-node9    -           generic   Running   tcp://192.168.61.54:2376    swarm-node0                v18.06.1-ce
yama@testbed1-yokosuka1:~$
```

# 実験利用
# X台のルータの直結接続環境



# 実験利用
# 環境構築(概要)

- ネットワーク作成
  - Overlayドライバの利用
  - ネットワークで利用するIPアドレスサブネットを指定
    - (指定なしの場合は、自動割当がされる)
- dockerイメージ(vyos)の起動
  - 接続ネットワーク、IPアドレス指定(1NICのみ)
  - 特権付与(ルーティングのため)
  - カーネルモジュールフォルダ共有
- vyosイメージへの追加操作
  - ネットワークの追加(IPアドレス指定)
  - カスタムbgpd.confの読み込み

# 実験利用
# 動作例(1/2) 経路の確認

```
[yama@testbed1-yokosuka1:~$ docker exec -it vyos-57 vtysh "-c sh ip bgp"
BGP table version is 0, local router ID is 0.0.0.57
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
*> 10.199.34.0/24   10.199.56.2                          0 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 i
*> 10.199.35.0/24   10.199.56.2                          0 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 i
*> 10.199.36.0/24   10.199.56.2                          0 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 i
*> 10.199.37.0/24   10.199.56.2                          0 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 i
*> 10.199.38.0/24   10.199.56.2                          0 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 i
*> 10.199.39.0/24   10.199.56.2                          0 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 i
*> 10.199.40.0/24   10.199.56.2                          0 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 4
*> 10.199.41.0/24   10.199.56.2                          0 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 i
*> 10.199.42.0/24   10.199.56.2                          0 56 55 54 53 52 51 50 49 48 47 46 45 44 43 i
*> 10.199.43.0/24   10.199.56.2                          0 56 55 54 53 52 51 50 49 48 47 46 45 44 i
*> 10.199.44.0/24   10.199.56.2                          0 56 55 54 53 52 51 50 49 48 47 46 45 i
*> 10.199.45.0/24   10.199.56.2                          0 56 55 54 53 52 51 50 49 48 47 46 i
*> 10.199.46.0/24   10.199.56.2                          0 56 55 54 53 52 51 50 49 48 47 i
*> 10.199.47.0/24   10.199.56.2                          0 56 55 54 53 52 51 50 49 48 i
*> 10.199.48.0/24   10.199.56.2                          0 56 55 54 53 52 51 50 49 i
*> 10.199.49.0/24   10.199.56.2                          0 56 55 54 53 52 51 50 i
*> 10.199.50.0/24   10.199.56.2                          0 56 55 54 53 52 51 i
*> 10.199.51.0/24   10.199.56.2                          0 56 55 54 53 52 i
*> 10.199.52.0/24   10.199.56.2                          0 56 55 54 53 i
*> 10.199.53.0/24   10.199.56.2                          0 56 55 54 i
*> 10.199.54.0/24   10.199.56.2                          0 56 55 i
*> 10.199.55.0/24   10.199.56.2             1            0 56 i
*  10.199.56.0/24   10.199.56.2             1            0 56 i
*>                  0.0.0.0                  1        32768 i
*> 10.199.57.0/24   0.0.0.0                  1        32768 i

Total number of prefixes 24
yama@testbed1-yokosuka1:~$
```

```
[vyos# sh ip route
Codes: K - kernel route, C - connected, S - static, R - RIP, O - OSPF,
       I - ISIS, B - BGP, > - selected route, * - FIB route

K>* 0.0.0.0/0 via 172.19.0.1, eth1
B>* 10.199.34.0/24 [20/0] via 10.199.56.2, eth2, 19:04:27
B>* 10.199.35.0/24 [20/0] via 10.199.56.2, eth2, 19:04:27
B>* 10.199.36.0/24 [20/0] via 10.199.56.2, eth2, 19:04:27
B>* 10.199.37.0/24 [20/0] via 10.199.56.2, eth2, 19:04:27
B>* 10.199.38.0/24 [20/0] via 10.199.56.2, eth2, 19:04:27
B>* 10.199.39.0/24 [20/0] via 10.199.56.2, eth2, 19:04:27
B>* 10.199.40.0/24 [20/0] via 10.199.56.2, eth2, 19:04:27
B>* 10.199.41.0/24 [20/0] via 10.199.56.2, eth2, 19:04:57
B>* 10.199.42.0/24 [20/0] via 10.199.56.2, eth2, 19:05:27
B>* 10.199.43.0/24 [20/0] via 10.199.56.2, eth2, 19:05:27
B>* 10.199.44.0/24 [20/0] via 10.199.56.2, eth2, 19:05:27
B>* 10.199.45.0/24 [20/0] via 10.199.56.2, eth2, 19:05:27
B>* 10.199.46.0/24 [20/0] via 10.199.56.2, eth2, 19:05:27
B>* 10.199.47.0/24 [20/0] via 10.199.56.2, eth2, 19:05:27
B>* 10.199.48.0/24 [20/0] via 10.199.56.2, eth2, 19:05:27
B>* 10.199.49.0/24 [20/0] via 10.199.56.2, eth2, 19:05:57
B>* 10.199.50.0/24 [20/0] via 10.199.56.2, eth2, 19:05:57
B>* 10.199.51.0/24 [20/0] via 10.199.56.2, eth2, 19:05:57
B>* 10.199.52.0/24 [20/0] via 10.199.56.2, eth2, 19:05:57
B>* 10.199.53.0/24 [20/0] via 10.199.56.2, eth2, 19:06:27
B>* 10.199.54.0/24 [20/0] via 10.199.56.2, eth2, 20:08:28
B>* 10.199.55.0/24 [20/1] via 10.199.56.2, eth2, 20:08:28
C>* 10.199.56.0/24 is directly connected, eth2
```

```
[vyos# traceroute 10.199.34.2
traceroute to 10.199.34.2 (10.199.34.2), 30 hops max, 60 byte packets
 1  10.199.56.2 (10.199.56.2)  0.532 ms  0.449 ms  0.299 ms
 2  10.199.55.2 (10.199.55.2)  0.700 ms  0.784 ms  0.681 ms
 3  10.199.54.2 (10.199.54.2)  1.097 ms  1.511 ms  1.417 ms
 4  10.199.53.2 (10.199.53.2)  1.862 ms  1.758 ms  1.904 ms
 5  10.199.52.2 (10.199.52.2)  2.160 ms  2.450 ms  2.595 ms
 6  10.199.51.2 (10.199.51.2)  3.073 ms  2.237 ms  2.930 ms
 7  10.199.50.2 (10.199.50.2)  3.291 ms  3.400 ms  4.161 ms
 8  10.199.49.2 (10.199.49.2)  4.433 ms  5.524 ms  5.410 ms
 9  10.199.48.2 (10.199.48.2)  5.754 ms  6.936 ms  6.783 ms
10  10.199.47.2 (10.199.47.2)  6.680 ms  7.176 ms  7.077 ms
11  10.199.46.2 (10.199.46.2)  9.605 ms  8.935 ms  8.801 ms
12  10.199.45.2 (10.199.45.2)  11.846 ms  11.263 ms  11.784 ms
13  10.199.44.2 (10.199.44.2)  12.330 ms  12.357 ms  11.827 ms
14  10.199.43.2 (10.199.43.2)  12.396 ms  12.762 ms  11.474 ms
15  10.199.42.2 (10.199.42.2)  14.297 ms  14.138 ms  13.900 ms
16  10.199.41.2 (10.199.41.2)  14.835 ms  13.227 ms  13.093 ms
17  10.199.40.2 (10.199.40.2)  14.812 ms  13.587 ms  13.428 ms
18  10.199.39.2 (10.199.39.2)  13.241 ms  14.157 ms  14.944 ms
19  10.199.38.2 (10.199.38.2)  14.210 ms  13.922 ms  13.786 ms
20  10.199.37.2 (10.199.37.2)  15.488 ms  13.332 ms  14.418 ms
21  10.199.36.2 (10.199.36.2)  18.534 ms  17.688 ms  18.624 ms
22  10.199.35.2 (10.199.35.2)  20.312 ms  21.803 ms  20.302 ms
23  10.199.34.2 (10.199.34.2)  20.641 ms  20.540 ms  19.847 ms
vyos#
```

# 実験利用
# 動作例(2/2) 流量測定

```
yama@testbed1-yokosuka1:~$ docker run -it --name iperf-1 --net=internal-24  --privileged -v /lib/modules:/lib/modules --env="constraint:node==swarm-node4" networkstatic/iperf3 -s
ERRO[0004] error getting events from daemon: EOF
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------

Accepted connection from 10.199.78.3, port 54128
[  5] local 10.199.24.4 port 5201 connected to 10.199.78.3 port 54132
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-1.00   sec  3.57 MBytes  30.0 Mbits/sec
[  5]   1.00-2.00   sec  4.46 MBytes  37.4 Mbits/sec
[  5]   2.00-3.00   sec  4.36 MBytes  36.6 Mbits/sec
[  5]   3.00-4.00   sec  3.93 MBytes  32.9 Mbits/sec
[  5]   4.00-5.00   sec  3.88 MBytes  32.5 Mbits/sec
[  5]   5.00-6.00   sec  4.58 MBytes  38.4 Mbits/sec
[  5]   6.00-7.00   sec  4.49 MBytes  37.7 Mbits/sec
[  5]   7.00-8.00   sec  4.63 MBytes  38.9 Mbits/sec
[  5]   8.00-9.00   sec  4.77 MBytes  40.0 Mbits/sec
[  5]   9.00-10.00  sec  5.19 MBytes  43.5 Mbits/sec
[  5]  10.00-10.06  sec   516 KBytes  69.2 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  5]   0.00-10.06  sec  45.3 MBytes  37.8 Mbits/sec    1             sender
[  5]   0.00-10.06  sec  44.4 MBytes  37.0 Mbits/sec                  receiver
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

iperf
サーバ

```
yama@testbed1-yokosuka1:~$ docker exec -it --privileged  iperf-2 /bin/sh
# ip route
default via 172.19.0.1 dev eth1
10.199.78.0/24 dev eth0  proto kernel  scope link  src 10.199.78.3
172.19.0.0/16 dev eth1  proto kernel  scope link  src 172.19.0.27
# ip route delete default
# ip route add default via 10.199.78.2
# ping 10.199.24.4
PING 10.199.24.4 (10.199.24.4): 56 data bytes
64 bytes from 10.199.24.4: icmp_seq=0 ttl=10 time=28.171 ms
^C--- 10.199.24.4 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max/stddev = 28.171/28.171/28.171/0.000 ms
# iperf3 -c 10.199.24.4
Connecting to host 10.199.24.4, port 5201
[  4] local 10.199.78.3 port 54132 connected to 10.199.24.4 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec  3.99 MBytes  33.4 Mbits/sec    1    122 KBytes
[  4]   1.00-2.00   sec  4.42 MBytes  37.0 Mbits/sec    0    147 KBytes
[  4]   2.00-3.00   sec  4.54 MBytes  38.1 Mbits/sec    0    168 KBytes
[  4]   3.00-4.00   sec  3.80 MBytes  31.9 Mbits/sec    0    184 KBytes
[  4]   4.00-5.00   sec  3.93 MBytes  32.9 Mbits/sec    0    199 KBytes
[  4]   5.00-6.00   sec  4.78 MBytes  40.1 Mbits/sec    0    216 KBytes
[  4]   6.00-7.00   sec  4.42 MBytes  37.0 Mbits/sec    0    277 KBytes
[  4]   7.00-8.00   sec  4.91 MBytes  41.2 Mbits/sec    0    362 KBytes
[  4]   8.00-9.00   sec  4.84 MBytes  40.6 Mbits/sec    0    474 KBytes
[  4]   9.00-10.00  sec  5.70 MBytes  47.8 Mbits/sec    0    617 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  45.3 MBytes  38.0 Mbits/sec    1             sender
[  4]   0.00-10.00  sec  44.4 MBytes  37.2 Mbits/sec                  receiver

iperf Done.
# yama@testbed1-yokosuka1:~$
```

iperf
クライアント

4VMに計54ホップを作成
iperf3 (デフォルト、TCP 10秒)で約40Mbps

(実際には100ルータ99ホップ作成を意図したが、
エラーにて、最長実用区間は54ホップ)

# 実験利用
# トラブル発生メモ

- 各種dockerコマンドの失敗
  - CPU負荷が高くなると高確率でエラー発生
    - インスタンス、ネットワークの作成失敗
    - 「ゆっくり」実施してエラー回避を
  - 環境構築時のエラーハンドリング
    - 依存関係を整理しやすくするため、IPアドレス自動設定は極力使わない
    - 重大エラーではプログラムを停止させる
    - (可能なものは)リトライをさせる
  - 例)
    - vyos 100インスタンス起動およびネットワーク等設定 = 約1時間で実施、エラー10程度
- まれに agetty プロセスが高負荷を継続
  - docker のバグの模様
  - pkill で対処
    - dockerインスタンス含め他への影響無し
- overlayが通らない
  - BUM(Broadcast Unkown unicast Multicast)抑止が関係?
  - 拠点間でのBUM
    - 活用研究会「資源」の仕様か?
    - 今回はoverlayドライバ利用を優先した為、単一拠点のみを利用
- overlay上の接続でARPを除くBUMが通らない
  - docker network overlayドライバの仕様
    - vxlanの仕様ではない
  - ARPのみproxy arpしている
  - 設定コスト節約のため、Dynamic RoutingでOSPFを使いたかった
    - non broadcast mode, neighbor指定で「たまに」動く
    - 動作優先のため、(unicast/TCPを利用する)BGPをつかうことに。

# 仮想化(docker)での対応結果

- 解決
  - 台数(及び、台数に付随して増加する管理コスト)
  - パス構成(L3のみ)
  - 共用利用
- 未解決
  - 遅延(別途、遠隔パス利用や、遅延フィルタを利用すればよい)
- その他
  - マルチテナント性確保は商用版(docker-ee)にて
  - VM性能に応じたインスタンス割当をすべきだった
  - 試験利用として、Data/Controlを分けない構成とした
  - 枯れてくればdocker/docker-machineのエラーは減るかもしれない
  - 「dockerで万事解決」ではない
    - 場合によりその他の仮想化(LXD等)や,ホスト側の機能の利用を
    - 特性を把握し、長所を活かした利用をすべき
- docker利用での副作用
  - dockerd等のプロセス生成
  - group:dockerの追加
  - 操作ユーザをsudoへ追加
  - ネットワーク操作に応じ、仮想ネットワークインターフェースが追加削除される

# 所感等

- 活用研究会について
  - 申請
    - (ほぼ手続き無しで)使い出せるのは、実験イメージをすぐに具現化できて助かる
  - 資源
    - 仮想化利用等で工夫をすれば、トポロジ等の制約はなくなる
    - VM台数やCPU割当等、高いスペックは(従来の) 申請対応と割り切る
  - インターネット接続
    - 1日2時間まで。前日までのメール申請。
      - 「手元」との違いがあり、試行錯誤は必須
      - インターネット接続があると作業が捗る
    - (今回は特別扱い(利用時間を実質無制限、平日即時対応)を頂いた)
      - =>利用/管理しやすいスキームがあれば。
- dockerについて
  - ポータビリティの高さ、ツール、イメージの豊富さは秀逸
  - (今回の用途では)負荷問題は、作成時に発生(動作時は比較的安定)
  - 隔離環境では、プライベートリポジトリを利用すべき
- その他
  - 今回のトポロジ特定の事象
    - 特定パスを複数の論理パスが同時に利用するため、<br>少量のバーストが増幅されて崩壊する可能性があった。
    - あらかじめの流量制限で緩和できるかもしれない

# 引用/参考

- Docker document
  - Docker Machine
    - https://docs.docker.com/machine/
  - Multi-host networking with standalone swarms
    - https://docs.docker.com/network/overlay-standalone.swarm/
  - Multiple Docker Networks
    - https://success.docker.com/article/multiple-docker-networks
- Github docker/labs
  - Overlay Driver Network Architecture
    - https://github.com/docker/labs/blob/master/networking/concepts/06-overlay-networks.md
- Vyos
  - 公式
    - https://vyos.io/
  - dockerイメージ(非公式)
    - https://hub.docker.com/r/2stacks/vyos/
  - Vyos users Japan
    - http://www.vyos-users.jp/