仮想環境におけるルータ構築省力化と通信性能評価 – VyOS, VirtualBoxを用いて –

黒瀬　浩[1,a)]　中沢　実[1]

概要：本稿では，仮想環境におけるルータ構築省力化と通信性能評価について述べる．従来，通信性能の問題から経路長を短くするネットワーク構成が望まれてきたが，SDN や通信速度向上により同一ホストを複数回巡回する処理も考えられるようになった．今後は経路長の大きなネットワークの利用が促進される可能性があり経路長の大きな構成を評価する．ルータに VyOS を，仮想環境に VirtualBox を用いて仮想環境上に多数のルータを直列に接続したネットワークを構築した．8CPU 3.7GHz のプロセッサによる評価環境では，96 の仮想マシンが稼働すること，40 ホップで 20Mbps，80 ホップで 7Mbps の性能が得られること，多数のルータを容易に構築できる自己設定が機能することを確認した．

1. はじめに

仮想化技術が進展し，ネットワークの評価が 1 台の物理マシンで容易に行えるようになった．従来，仮想環境でのネットワーク検証は，機能面が中心であったが，ハードウェア資源の高性能化から性能の検証も可能となった．クラウドサービスでは，高性能の物理マシンを多数用いて実用速度を実現している．

本稿では，入手が容易なデスクトップ PC やオープンソースソフトウェアを用いて，多数のルータによるネットワークを仮想環境上に構築し，通信性能の評価を行ない利用可能なネットワーク規模を確認する．また，多数のルータの構築を省力化する方法を提案する．

評価は，性能と構築性により行う．性能評価は，仮想環境上の通信性能とルータ起動時間，仮想マシン操作時間を対象とする．構築性評価は，多数のルータ構築の省力化を対象とする．

2 節で仮想環境のネットワークの関連研究を，3 節で検証のモデルと項目を，4 節でルータの自動設定方法を，5 節で検証環境を，6 節で結果と解析を，7 節でまとめを述べる．

2. 関連研究

ルータの性能比較 [1] では，Cisco 社専用機とオープンソースルータ Vyatta の性能を比較している．ここでは，Vyatta は VirtualBox[2] 環境に構築している．Vyatta は現在，Brocade 社により管理されているが，派生したオープンソース版の VyOS[3] が利用できる．

コントローラから仮想環境上の仮想マシン (VM) の複製，起動を行い，多数の VM の実行結果を 1 つの VM に収集する自動実験システムがを提案され 90 以上の VM を用い性能評価が行われている [4]．

VirtualBox は，VM のスナップショットや，別の物理マシンへの移行，グループごとの管理など機能が多く，GUI と CUI による管理が利用できる．

以上から VyOS および VirtualBox を用いて検証を行う．

3. 検証

ネットワークの高速化と新たな分散処理形態から経路長の大きなネットワーク構成の利用が考えられるため，経路長の大きなネットワークを評価する．

L3 スイッチのスイッチング性能や LAN 通信速度向上により通信性能に対する経路長の影響が低くなった．SDN(Software Defined Network) の代表的な OpenFlow プロトコル [5] では，MAC アドレスや IP アドレスをコントローラが書き替えて通信経路を柔軟に制御することができる．ビッグデータの処理に MapReduce などの再帰処理が用いられている．MapReduce の並列実行処理の例としてマルチコアによる MapReduce の評価が行われている [6]．ここで map ステージと reduce ステージを各コアで分散処理している．map 用ホスト，reduce 用ホストを用意して再帰的処理を行う場合には，map ホスト，recude ホストをパケットが循環する．この循環処理をネットワーク制御で行うとアプリケーションプログラムは RPC(Remote

[1] 金沢工業大学工学部
Kanazawa Institute of Technology, Hakusan-shi, Ishikawa 924–0834, Japan
[a)] kurose@neptune.kanazawa-it.ac.jp

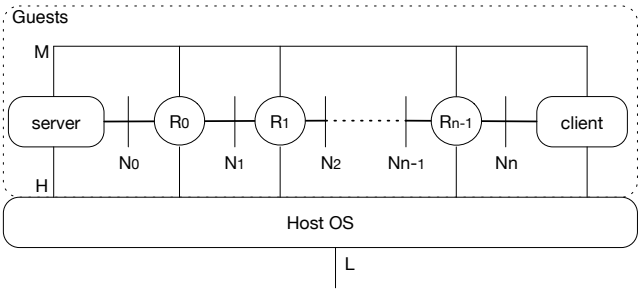

図 1 検証モデル
Fig. 1 Verification model

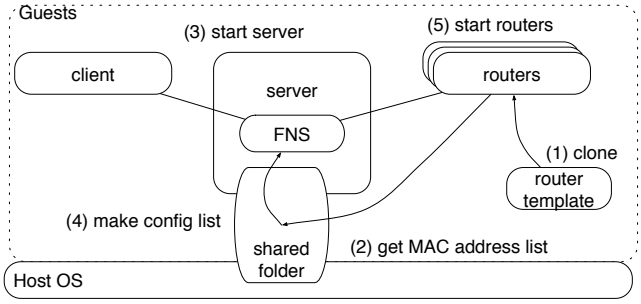

図 2 ルータ ID の特定方法
Fig. 2 Specific method of router ID

Procedure Call)[7] のように処理を依頼して応答を待つことで，プログラム処理の負担軽減が期待できる．ホストの循環をレイヤ 3 で制御する場合，経路長が大きなネットワーク構成の通信が必要となる．

検証のモデルと項目を述べる．

3.1 モデル

図 1 に検証するネットワークのモデルを示す．1 台の物理マシン上に検証するネットワークを構築する．ネットワーク環境は，サーバ server, クライアント client, ルータ R_x の各機器から成る．サーバ・クライアント間に n 台のルータを設けて，n+1 個のサブネットによる直列接続を構成する．

性能検証の対象は，サーバ・クライアント間を n 台のルータで接続した N_0 から N_n のまで経路で，通信範囲はゲスト OS 間のみである．

ルータ, クライアントは, 保守回線 M を介してサーバと通信することができる．サーバに DHCP(Dynamic Host Configuration Protocol) および NFS(Network File System) のサーバ機能を設け，保守回線のみで動作させる．この回線は，ルータの初期設定および，測定結果の取得に用い，ゲスト OS 間のみの通信が可能である．

各機器からホスト OS への回線 H は，NAT(Network Address Translation) 接続によりホストを経由して物理インターフェイス L を介してインターネットに接続される．この回線は，NTP (Network Time Protocol) による時刻同期，DNS(Domain Name System) による名前解決，または，追加ソフトウェアの入手用に用い，ゲスト間での通信は行わない．

検証ネットワークの前提事項を説明する．

検証構成では，以下の制約を解除し，最大 254 ホップの通信を目指す．障害を起こしたルータの検出を容易とするために動的ルーティングを用いるが RIP(Routing Information Protocol) の最大ホップ数は 14 のため，OSPF(Open Shortest Path First) を採用する．使用する OS の最大 TTL(Time to Live) は 64 に制限されているため値を変更する．

利用する VM の NIC(Network Interface Controller) でゲスト OS 間でのみ通信できる “内部ネットワーク” には，サブネットに応じた識別名を設定する．これは，VM 管理側では通信可能範囲を内部ネットワーク識別名を用いていることによる．

3.2 検証項目

データセンタなどの特別な設備を必要としない一般的な 1 台の物理マシン上の仮想環境で，実現可能なネットワーク規模を確認する．

ルータを直列に接続したネットワークにおいて，通信性能，ルータ起動時間，複数の仮想マシンの操作性，および，多数のルータ構築の省力化について検証する．

利用資源は，ホスト OS で消費するメモリ資源および CPU 使用率を確認をする．

性能評価は，サーバ・クライアント間の通信性能，パケット廃棄率，各ルータの起動時間，最後に起動したルータがログイン可能となるまでの時間，サーバのルーティングテーブルに全ルータが反映されるまでの時間，および，VM 操作時間による．通信性能評価は，ICMP，TCP，UDP の各プロトコルを用いる．

VM 操作性の評価は，VM の複製（クローン），起動，停止，削除，および，設定変更の各時間を用いる．

起動時間や経過時間は，各ルータが未設定の場合と設定済みの場合を取得する．

4. ルータ仮想マシンの自動設定方法

サーバ VM とクライアント VM は台数が各 1 台であり個別に構築可能であるが，ルータは多数となるため VM の複製と設定の自動化が望まれる．各ルータが起動時に自己設定する方法では，自身の ID 特定と設定情報の取得が必要である．ホスト OS 側とルータ OS で共通に得られる項目に NIC の MAC アドレスがあり，ルータ ID 特定に用いる．

ルータ ID 特定とルータ起動時の自己設定の方法について述べる．

4.1 ルータ ID の特定

ルータ ID の特定方法を図 2 に示す．

手順 0 テンプレートの作成：テンプレートとなるルータ VM を作成し，CPU 数，CPU 使用率制限，メモリ容量，ディスク容量，NIC 数と種類などの論理ハードウェア資源を VM 管理ツールより設定する．ルータの OS である VyOS をインストールし，以下の設定を行う．NIC1 はホストとの NAT 接続に，NIC4 はサーバとの保守回線に，それぞれ DHCP で IP アドレスを取得し接続する．サーバと NFS でファイルを共有する．起動時にルータの設定を行うスクリプトを自動実行する．

手順 1 ルータ VM の複製：必要なルータ数のクローンをテンプレートから VM 管理ツールを用い複製する．この時，VM 管理ツールにより各 NIC に独立した MAC アドレスが割り当てられる．

手順 2 VM 情報の設定と取得：複製した各 VM の NIC の種類および ID，VM の所属グループを VM 管理ツールにて設定する．NIC2 と NIC3 のネットワーク識別名は，ルータ番号からサブネット番号を生成し設定する．保守回線に接続される NIC4 の MAC アドレスを取得して VM 名と NIC4 の MAC アドレスから成る一覧を作成する．

手順 3 サーバ VM の起動：サーバを起動しホスト OS のファイルを共有フォルダをサーバから利用可能とする．

手順 4 ルータ設定情報の作成：共有フォルダの MAC 一覧から各 VM の MAC アドレス，ホスト名，サブネット番号から成る設定一覧を作成し，NFS 管理下のディレクトリに格納する．

手順 5 ルータ VM の起動：起動するルータ数を指定してルータを VM 管理ツールより起動する．各ルータは起動時に自身が設定済みか調べる．未設定の場合は，NFS マウントしたディレクトリの設定一覧ファイルから自身の設定情報を取得し，ルータ設定ファイルを作成して実行する．設定済みを示すファイルが存在すれば，設定は行わない．起動後，起動時間を NFS ディレクトリのファイルに書く．

4.2 ルータの自己設定

ルータ起動時の処理概要を Algorithm 1 に示す．

ルータは保守回線を通じてサーバより IP アドレスを DHCP により取得する．保守回線を通じてサーバと NFS によりファイルを共有する．ルータが設定済みであるかファイルの存在により判断する．未設定の場合は，保守回線の MAC アドレスを調べ，NFS ディレクトリにある設定ファイルから該当する MAC アドレスが存在する行を得る．その行には，ホスト名，サーバ方向のサブネット番号，クライアント方向のサブネット番号があるので，その情報を元にルータ設定ファイルを作成する．ルータ設定ファイルを実行して成功すれば，設定済みファイルを作成する．最後に起動時間の情報を NFS ディレクトリのファイルに書く．

Algorithm 1 Setup at startup

```
 1: getIPaddrByDHCP( NIC4 )
 2: mountFNSdir()
 3: if ¬exists( DoneFile ) then
 4:    MACaddr ⇐getMacAddr( NIC4 )
 5:    makeConfig( getConfigInfo( MACaddr ) )
 6:    if executeConfig() is Success then
 7:       makeDoneFile()
 8:    end if
 9: end if
10: writeStartupInfo()
```

ルータの設定ファイルの主要部分をリスト 1 に示す．`$HOST`, `$NS`, `$NC` は，それぞれホスト名，サーバ方向のサブネット番号，クライアント方向のサブネット番号に置換される．ホスト名は，VM 名を用いた．VM 名の右 3 桁はルータ番号である．linux の LAN インターフェイス eth0 から 3 は，VirtualBox の論理ネットワーク NIC1 から 4 にそれぞれ対応している．設定では，ホスト名，各 NIC アドレスの設定と動的ルーティングの設定を行う．

リスト 1 ルータ設定スクリプト
List 1 Router config script

```
#/bin/vbash
source /opt/vyatta/etc/functions/script-template
configure
load /opt/vyatta/etc/config.boot.default
commit
delete system console device ttyS0
set system host-name $HOST
set system time-zone Asia/Tokyo
set system ntp server ntp.nict.jp
set service ssh port 22

set interfaces ethernet eth0 address dhcp
set interfaces ethernet eth1 address 172.16.$NS.1/24
set interfaces ethernet eth2 address 172.16.$NC.2/24
set interfaces ethernet eth3 address dhcp

set protocols ospf parameter router-id 172.16.$NS.0/24
set protocols ospf network 172.16.$NS.0/24
set protocols ospf network 172.16.$NC.0/24
set protocols ospf area 0.0.0.0 network 172.16.$NS.0/24
set protocols ospf area 0.0.0.0 network 172.16.$NC.0/24
set protocols ospf redistribute connected

commit
save
exit
```

5. 検証環境

ホスト OS に linux を，仮想環境は VirtualBox を，ルータは VyOS を用いる．物理マシンは，ATX(Advanced Technology eXtended) サイズのデスクトップ PC を用いる．

検証環境の構成を表 1 に示す．物理マシンの CPU 数は，4 コア 8HT(Hyper Threading) で，OS から見て 8CPU と

表 1 検証環境構成
Table 1 Specification

item\ machine	Host OS	server	client	router
CPUs	8	1	1	1
CPU clock	3.7GHz	100%	100%	100%
memory	64GB	1GB	1GB	256MB
storage	2TB(raid1)	10GB	10GB	2GB
NIC1	L	H	H	H
NIC2	–	N_0	N_x	N_n
NIC3	–	–	–	N_{n+1}
NIC4	–	M	M	M
OS	ubuntu	ubuntu	ubuntu	vyos
routing	–	OSPF	OSPF	OSPF
shared file	Yes	Yes	–	N/A
NFS	–	server	client	client
H DHCP	server	client	client	client
M DHCP	–	server	client	client

なる．VM には，各 1CPU 割り当てる．VM の CPU clock は，VM 管理ツールで設定した CPU 使用率制限を記載している．VyOS の稼働推奨メモリは 512MB であるが，使用量が 170MB 程度であるため 256MB を割り当てた．物理ストレージは，ミラーリング構成ハードディスクを用いた．表 1 の NICn の L, H, N_x, M は，それぞれ図 1 の物理 LAN 回線，VM からホスト OS への NAT 回線，サーバ VM・クライアント VM を接続する検証ネットワーク，ルータ設定用のメンテナンス回線を意味する．サーバ VM の NIC2, クライアント VM の NIC4, ルータ VM の NIC2 から 4 はネットワーク識別名を表す．クライアントの NIC2 の識別名 N_x は，サーバと同一のサブネット N_0 からルータ数+1 まで検証条件に応じて変える．H DHCP はゲスト OS からホスト OS へのホスト回線 H の，M DHCP はゲスト OS 間のメンテナンス回線 M の，DHCP 機能の役割を意味する．

ソフトウェアのバージョンを以下に示す．

ホスト OS のソフトウェアは, Ubuntu amd64 15.04(kernel 3.13.0) および VirtualBox 4.3.2 を, サーバおよびクライアントの OS は，Ubuntu amd64 15.04(kernel 3.13.0) を，ルータの OS は，vyos amd64 1.1.5 を用いた．

サーバおよびクライアントのルーティング機能は，quagga(zebra, opsfd) を，サーバの DHCP サーバ機能は，isc-dhcp-server を，サーバの NFS サーバ機能は，nfs-kernel-server を用いた．

設定ファイルの作成には bash, sed, awk 等 Linux コマンドを用いた．複数ルータの管理は，VirtualBox のコマンドラインインターフェースを利用して bash スクリプトファイルを作成した．

6. 結果と解析

評価結果を利用資源と性能から述べる．

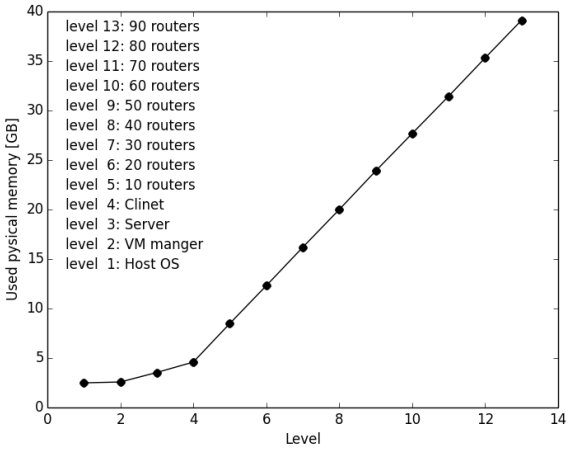

図 3 消費物理メモリ
Fig. 3 Used physical memory

6.1 利用資源

ホスト OS で消費した物理メモリ量と CPU 使用率を測定した．

図 3 に消費した物理メモリ量を示す．各 OS の消費メモリ量は，top コマンドにてホスト OS 側で測定した．

ホスト OS 起動に 2.5GB 消費し，VirtualBox 管理ツールを起動すると 2.57GB となった．次に，サーバ VM 起動後 3.53GB となり，クライアント VM 起動後 4.58GB となった．その後，ルータ VM を増分を 10 として増加させると 1VM あたり平均 0.38GB 増加した．ルータ VM には，論理メモリを 256MB 割り当てたが，それより多くの物理メモリを消費した．VM に割り当てられたメモリ資源に加えて VirtualBox 管理領域及び Linux のプロセス管理用メモリが消費されることが原因と考えらる．

ルータ VM 数が 92 から 94 になった際に，物理マシンのメモリが空いているにも関わらずルータ VM は起動に失敗した．サーバ VM, クライアント VM 合わせて最大 96VM までの起動を確認した．各 VM 上のメモリ容量は，割り当てた論理メモリサイズであり，使用量はサーバ, クライアントとも約 860MB で，ルータは約 166MB であった．物理資源が足りる範囲で最大 96VM までの実行ができるが，状況によりそれ以下となる結果を得た．

VM インスタンス起動に失敗する場合は，VM 管理ツールからの起動では，`NS_ERROR_FAILURE(0x800004005)` が表示され，VM インスタンスのログファイルは作成されない．dmesg では，`vboxdrv: [アドレス] VMMR0.r0` が記録された．VirtualBox 管理モジュール内での不具合と考える．同一マシン上で，Ubuntu 15.04(kernel 3.19.0), VirtualBox 5.0.2 の環境では，ルータ VM 数は 104 まで生成できることを確認したが，物理メモリ上限に達する前に同様の現象となった．

次に CPU 使用率について述べる．

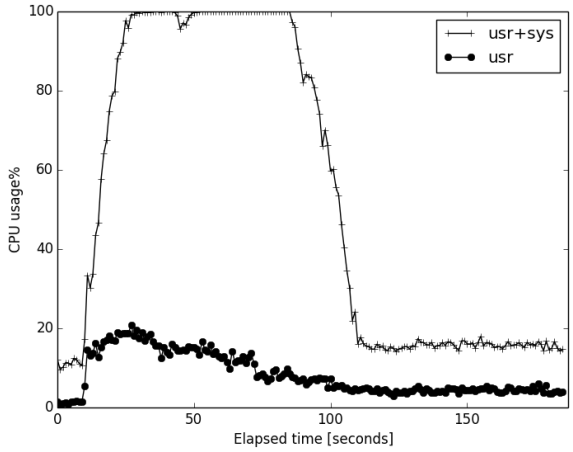

図 4 CPU 使用率（設定済み 30 台）
Fig. 4 Used CPU (30 configured routers)

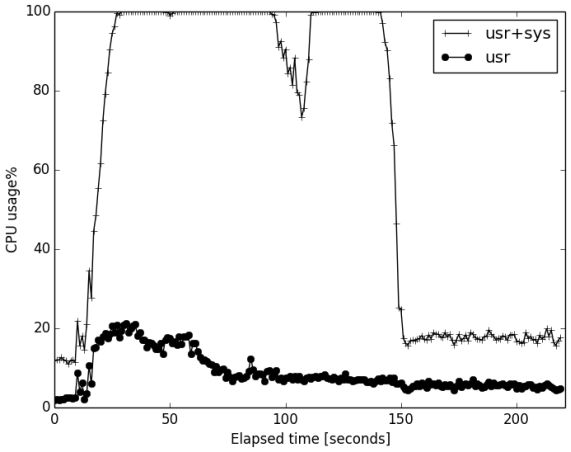

図 5 CPU 使用率（未設定 30 台）
Fig. 5 Used CPU (30 unconfigured routers)

図 4 は設定済みルータ 30 台同時起動時の，図 5 は未設定ルータ 30 台同時起動時の，図 6 は設定済みルータ 90 台同時起動時の経過時間による CPU 使用率で，開始 10 秒からルータ順次を起動している．計測は，iostat -c コマンドを用いて，1 秒間隔でユーザ空間 (usr) と OS 空間 (sys) の CPU 資料率を測定した．サーバ VM のルーティングテーブルに全ルータのネットワークが反映された時点で計測を終了した．

サーバ VM とクライアント VM を起動した時点で，CPU は 10%程度消費している．ルータ VM を起動し終えた後の定常状態では，30 ルータ VM で CPU 使用率は 20%程度，90 ルータ VM で 30%程度であった．ルータ VM 起動開始から CPU を 100%使用し，各ルータの OS 起動完了まで CPU 資源を消費する．ルータ VM 起動時の CPU 使用の大部分は OS 内の処理である．図 5 の 100 秒経過付近で CPU 使用率が 20 秒程度下がっている．多数のクローン作成など負荷が大きい場合に VM 管理ツールが 20 から 30 秒程度応答しない場合が観測された．VM 管理への負荷が多い場合にメモリ管理などで処理が遅延することが考えられる．未設定ルータを 40 台以上同時起動するとホスト側との NAT 接続の DHCP による IP アドレス取得に失敗して設定が完了しないルータ VM も見られた．VM 管理への過度な要求は動作が不安定となる．この処理遅延は，設定済み 90 台の同時起動では発生しておらず，ルータ設定処理が VM 管理へ与える負荷は高い． 図 7 に起動処理の時間と平均 CPU 使用率および起動後の平均 CPU 使用率を示す．ルータ VM 同時起動の際にホスト OS の CPU 使用率が定常状態から上がり起動が終わり安定するまでの期間を Busy Time とする．ルータ起動後の安定状態では，受信したパケットの処理やデーモンプロセスの処理が動作している．図 7 では，左軸に Busy Time の値を表している．設定済み 10 ルータで約 50 秒，90 ルータで 300 秒程度を要

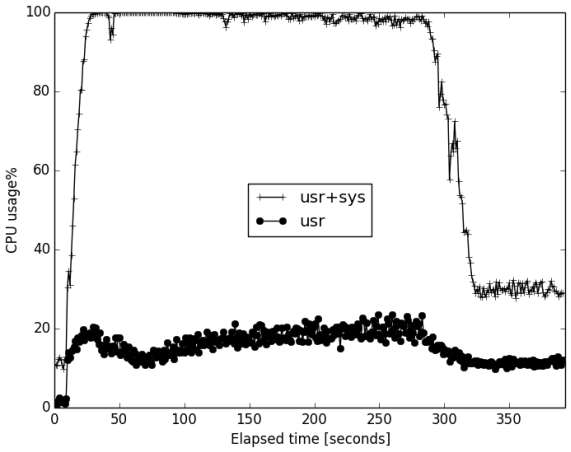

図 6 CPU 使用率（設定済み 90 台）
Fig. 6 Used CPU (90 unconfigured routers)

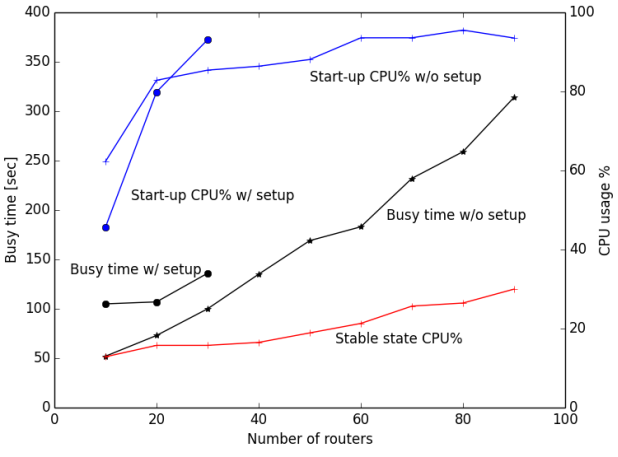

図 7 起動時間と CPU 使用率
Fig. 7 Busy time and CPU usage

した．ルータ起動処理の平均 CPU 使用率は，20 ルータ以上の VM 同時起動で 80%を超えた．ルータ VM 起動後の定常処理になった場合の CPU 使用率は 10 ルータ VM で

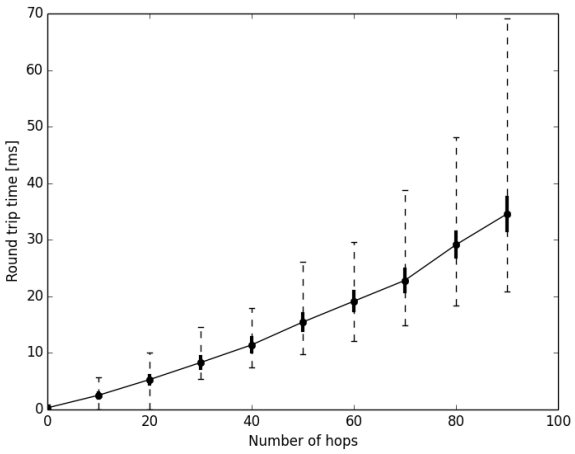

図 8 ICMP 通信性能
Fig. 8 ICMP round-trip time

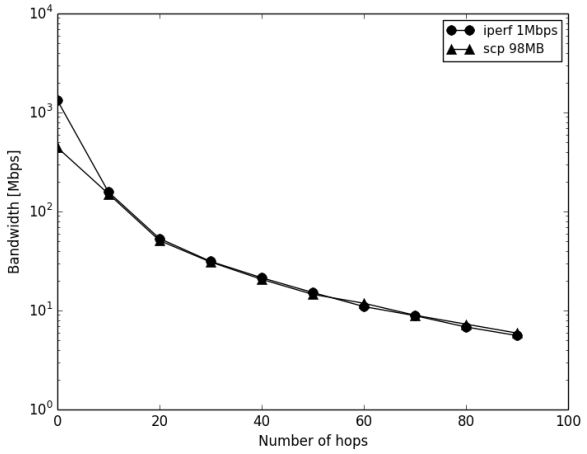

図 9 TCP 通信性能
Fig. 9 TCP performance

20%程度であるが，ルータ VM を増やすにつれ上昇し 90 ルータ VM では 30%程度となった．

多数の VM を 1 台の PC 上で実現する場合に CPU 性能と CPU 数に依存する．特に VM 管理への負荷が高いと図 5 のようにオーバーヘッドが生じ，ゲスト OS 上での性能評価やリアルタイム性に影響を与える．

6.2 ICMP 通信性能

1 秒間隔で 30 回 ping を実施するテストを 5 回繰り返し，平均，最大，最小，標準偏差を求める．

図 8 に ICMP の応答時間を示す．図では，点線で最小値および最大値の範囲を，太線で標準偏差の大きさを表す．いずれの場合も廃棄は発生しなかった．

応答時間は，ホップ数に対して比例して大きくなる．40 ホップで 10ms，70 ホップで 20ms 程度の結果を得た．ホップ数が増えるにつれ最小時間と最大時間の差も大きくなった．最悪値は，試験範囲では，70ms 程度であった．

国内サイトのアクセスで 10 ホップ程度，海外サイトで 20 から 30 ホップ程度であるが，仮想環境内の 30 ホップで 10ms 程度であるので，ホストマシンの CPU 負荷が高くなければ，仮想環境で現実環境の評価を代用することも可能と考える．

6.3 TCP 通信性能

測定は，iperf[8] により 1Mbps の負荷を 30 秒かける場合と，98MB のファイルを scp で転送する場合をそれぞれ 5 回行い平均を表示している．iperf での window サイズは標準値の 93.5KB である．

図 9 に TCP プロトコルを用いた場合の通信性能を示す．縦軸は対数表示である．

サーバとクライアントが同一サブネットに存在する 0 ホップでは，iperf で 1340Mbps, SCP で 443Mbps であったが 10 ホップ以上で iperf と scp で同様な性能となった．通信性能は，10 ホップで 160Mbps 程度，20 ホップで 53Mbps 程度，30 ホップで,31Mbps 程度，60 ホップで 10Mbps 程度で，90 ホップでは，5.5Mbps 程度であった．

6.4 UDP 通信性能

測定は，iperf で UDP プロトコルを用い負荷を 1Mbps, 10Mbps, 100Mbps, および 1Gbps に変化させ 30 秒間通信する．これを 5 回繰り返して平均を求める．バッファサイズは標準の 208KB を用いた．

図 10 から 12 に UDP プロトコルを用いた場合の通信性能，データグラム廃棄率，および，ジッタを示す．

1Mbps では 90 ホップまで，10Mbps では 70 ホップまで安定している．10Mbps で 80 ホップからデータグラム廃棄が発生している．100Mbps では 10 ホップ以降，1Gbps では，0 ホップすなわちサーバとクライアントが同一サブネットに存在する時から廃棄が発生した．100Mbps および 1Gbps では，廃棄率が大きくなったため，途中で計測を打ち切っている．同様に図 11 で廃棄率が大きな 100Mbps および 1Gbps は参考データとして表示している．廃棄率が 5%以内であれば，ジッタは 2ms 程度である．

6.5 平均ルータ起動時間

ルータの OS が起動してから必要なサービスプロセスを起動し終えるまでの時間を/proc/uptime から取得し，同時起動したルータ数の平均をとる．起動時にルータの自己設定を行う場合と設定済みの場合を比較する．

図 13 に平均起動時間を示す．図では同時起動したルータの最小時間および最大時間を縦点線で，標準偏差を縦太線で表示した．ルータが設定済みの場合は，同時起動するルータ台数が増えても安定した時間となっている．1 台のみ起動した場合，設定済みの場合は 23 秒程度，自己設定

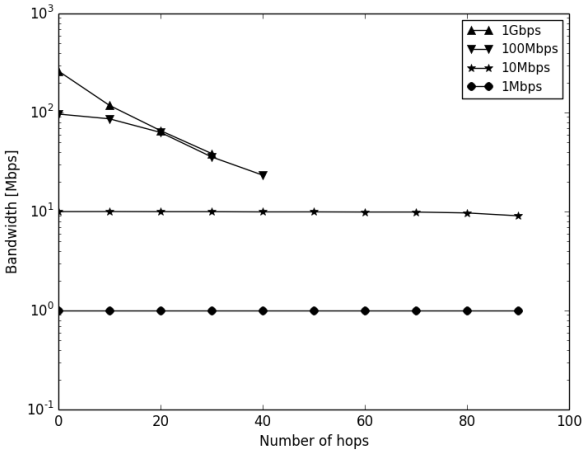

図 10 UDP 通信性能
Fig. 10 UDP performace

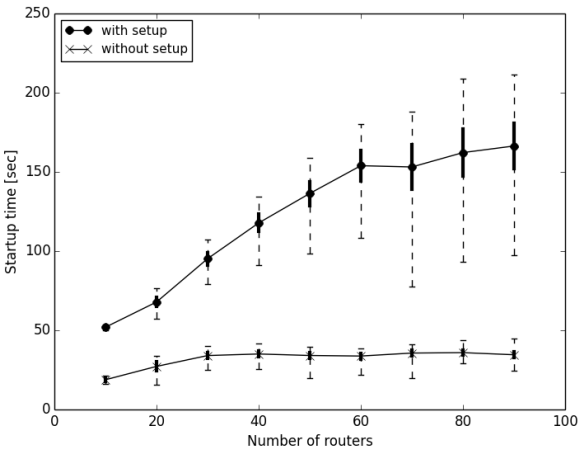

図 13 平均起動時間
Fig. 13 Average startup time

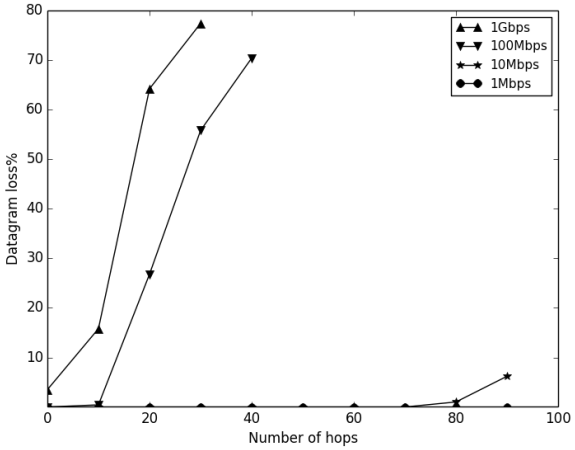

図 11 UDP 廃棄率
Fig. 11 UDP datagram loss%

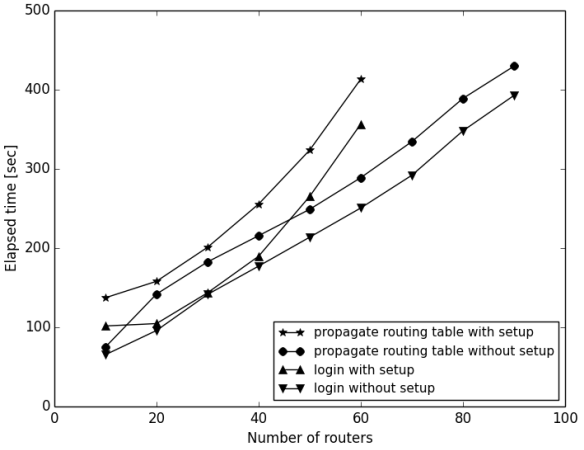

図 14 最終起動時間
Fig. 14 Startup time of latest router

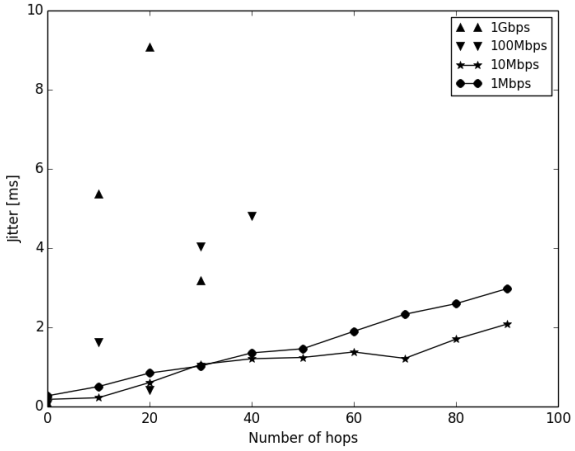

図 12 UDP 通信ジッタ
Fig. 12 UDP jitter

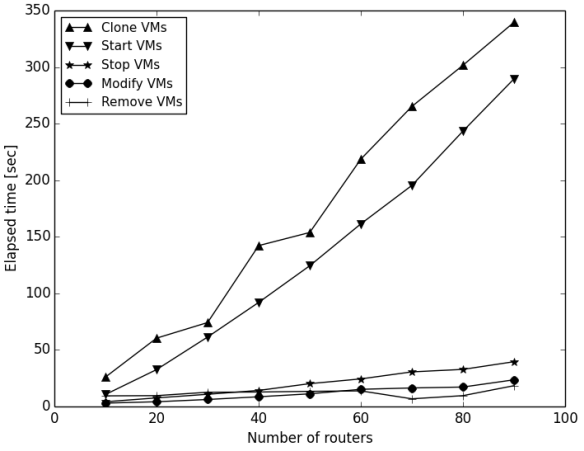

図 15 VM 数による管理ツール実行時間
Fig. 15 VM management performance

時は 26 秒程度であった．ルータ OS 起動時には，ブートローダ画面および OS 選択画面でそれぞれ 5 秒のキー入力待ち時間があり，設定済みの場合は，その間に他ルータの起動処理が行われ，物理 CPU の競合が少なかったためと考える．自己設定を行う場合 60 ルータ同時起動以降は平均起動時間の増加は少なくなるが，ルータが順次起動され

物理 CPU のリソース競合が軽減されたと考える．

6.6 全ルータの起動時間

図 14 に最初のルータを起動し始めてから，最後に起動されたルータがログイン可能となるまで，および，サーバのルーティングテーブルに全てのサブネットが反映されるまでの経過時間を示す．

自己設定する場合の 70 台同時起動以降は，ホスト OS 側の DHPC アドレス取得で失敗する場合があり不安定となったため結果の表示をしていない．設定済みの場合は，ログイン可能時間とルーティングテーブル反映完了までの差は 40 秒程度で，自己設定する場合は，60 秒程度であった．

6.7 VM 操作実行時間

図 15 に VM 管理ツールによる VM の複製，起動，停止，設定変更，および，削除に要した時間を示す．ホスト OS 側から見たルータ VM のイメージファイルのサイズは 383MB で，ミラーリングされたハードディスクに格納している．設定変更は，VM のグループおよび NIC 識別名を設定する．削除は，登録を除去するのみでなくイメージファイルも削除する．処理に時間を要するのは複製と起動で，複製は VM あたり 4 秒程度，起動は 2 秒程度要している．停止，設定変更および削除の処理時間は複製や起動に比べて少ない．

7. おわりに

1 台の物理マシンの仮想環境上に 90 台以上のルータから成るネットワークを構築した．ルータのテンプレートから起動時に自己設定することで多数のルータ構築の省力化と容易な構成変更の方法を提案した．ホップ数によるバンド幅や遅延時間を提示した．

デスクトップコンピュータによる仮想環境では，バンド幅，遅延時間の制約はあるが数十台規模のルータによるネットワークが構築できることを示した．多数のゲスト OS 起動など VM 管理への負荷が高い処理を行う場合には，ホスト CPU リソースに応じてゲスト OS の処理配分に注意が必要となる場合がある．

物理メモリに余裕があっても VM が起動できなくなる現象が発生した．カーネル内の VirtualBox ドライバで Linux カーネル内リソース取得に失敗していると考えられるが原因特定に至らなかった．今後，カーネル内の処理を確認したい．

従来の 30 ホップ以内のネットワーク設計に対して，SDN を用いた再帰処理で経路長の大きな通信の必要性を述べた．今後，ホストを循環するネットワーク制御方式とアプリケーションインターフェイスについて検討・設計する．

参考文献

[1] E. Guillen, A. M. Sossa, and E. P. Estupinan, “Performance Analysis over Software Router vs. Hardware Router: A Practical Approach,” Proceedings of the World Congress on Engineering and Computer Science, WCECS 2012, pp.973–978, Oct. 2012.

[2] Oracle VirtualBox, `https://www.virtualbox.org/`

[3] VyOS, `http://vyos.net/`

[4] N. Huber, M.v. Quast, F. Brosig, and S. Kounev, “Analysis of the Performance-Influencing Factors of Virtualization Platforms,” Proceedings of the On the Move to Measurement Systems, OTM 2010, pp.811–828, Oct. 2010.

[5] Open FLow Switch Specification Version 1.3.0, Open Networking Foundation, Jun. 2012.

[6] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis, “Evaluating MapReduce for Multi-core and Multiprocessor Systems,” High Performance Computer Architecture, 2007. HPCA 2007. IEEE 13th International Symposium, pp.13–24, Feb. 2007.

[7] RPC: Remote Procedure Call Protocol Specification Version 2, RFC1831, The Internet Engineering Task Force (IETF), Aug. 1995.

[8] Iperf - The TCP/UDP Bandwidth Measurement Tool, `https://iperf.fr/`