Vol.2018-CE-145 No.9
2018/6/9

論　文

仮想ネットワーク構築ライブラリ VITOCHA とネットワーク技術者教育

鈴木常彦[1,a)]

概要：FreeBSD の OS パーティショニング機能である Jail とネットワーク仮想化機能 VIMAGE を操作し、自由に仮想ネットワークをプログラミングできる Ruby ライブラリ VITOCHA を作成した。VITOCHA による仮想ネットワークは DNS キャッシュポイズニングの解明に大いに役立ったほか、ネットワーク技術者を集めた DNS 勉強会「DNS 温泉」においても VITOCHA による DNS シミュレーション環境は重要な役割を担っている。また本学においては遅延やパケットロスを変化させ TCP のスループットを測定する工学実験の授業に役立てている。これらの実践について VITOCHA の解説を交えて報告する。

キーワード：DNS,FreeBSD,jail,VIMAGE,VITOCHA, 仮想, ネットワーク,

Tsunehiko Suzuki[1,a)]

1. はじめに

技術者教育の現場において多数のネットワーク機器やサーバを揃えたうえ、多くの受講者を教育する環境を整えることは費用の面でもスペースの面でも運用の手間の面でも困難であり、大きな課題であった。

本論文ではパソコン上で自在に仮想ネットワークを構築することができる Ruby ライブラリを開発し、大学の授業やネットワーク技術者たちの勉強会で役立てている現状について報告する。

2. 類似システム

本報告で用いているネットワークシミュレーション環境に類したものとして代表的なものを以下に挙げる。

2.1 インターネットシミュレータ

2002 年から 2007 年頃にかけて筆者が製作していたインターネットシミュレータ [1] があった。1 台のラックに約 30 台のサーバを積み、それぞれ 3-4 台の VM を動作させて小規模なインターネットを模擬していた。今回報告するシステムはこの初代システムの仮想化をさらに進めて 1 台のパソコンで動作させられるようにしたものである。

2.2 IMUNES

ザグレブ大学で研究、提供されている IMUNES [2], [3] は筆者のシステムと同等の仮想ネットワーク構築が可能であり、さらに優れた GUI を持ち合わせている。ただし、筆者の場合は CUI 上の直感的なプログラミングにより仮想ネットワークを構築できることを設計思想としており、アプローチが異なっている。CUI のほうが高等教育やネットワーク技術者の現場においては扱いやすいと筆者は考えている。

[1] 中京大学
[a)] tss@suzuki.sist.chukyo-u.ac.jp

2.3 ns-3

ネットワークシミュレーションソフトウェアとして ns-3 [4] はとても有名であるが、数値シミュレーションであり、今回のように実パケットや種々のサーバソフトウェア実装を試すことのできるものではなく、用途の異なったものであるといえる。

2.4 PacketTracer

CNA(CISCO Network Academy) [5] で用いられている PacetTracer [6] は CISCO Systems 社 のルータ OS である IOS 学習用ソフトウェアである。IOS と同じコマンドが動作する仮想ルータを接続してネットワークシミュレーションが行える。ただしあくまでソフトウェアシミュレーションであり、ns-3 と同様に実パケットや種々のサーバソフトウェア実装を試すことはできない。

2.5 StarBED4

言わずと知れた StarBED4 [7] は他に類をみない大規模なネットワークシミュレータであるが、教育現場や個々人の技術者の手元で日々手軽に用いることはできず、ポータブルに用いることができる本報告の環境と比するものではないと考えている。

3. VITOCHA の構成と設計

VITOCHA は FreeBSD の jail を扱いやすくすることを目的に 2012 年から開発を行ってきたものである。Ruby によるプログラミングで仮想ネットワークを構築できるライブラリ群となっている。

3.1 jail と VIMAGE

FreeBSD には jail と VIMAGE という仮想化機構がある。jail は 1 つの FreeBSD を仮想的に複数の FreeBSD にパーティショニング (隔離) する技術である。ファイルシステム、プロセスなどを他の jail から隔離する。仮想マシン (VM) と異なりカーネルや主要なシステムファイルは共用するため少ないリソースで多数の jail を動作させることができる。VM だと数台しか動かないマシンで数十の jail を動作させられる。

VIMAGE は jail が提供する隔離機構に加えて、ネットワークスタックやルーティングテーブルなどの隔離も行ってくれる機構である。FreeBSD の ifconfig が提供する仮想インターフェイスや仮想ブリッジを VIMAGE で各 jail 内に隔離することにより、1 台のマシンの中で複数の jail を繋ぐ仮想ネットワークを構築することができる。

FreeBSD にはさらにネットワークスタック上でネットワーク特性を変化させる DUMMYNET と呼ばれる機構があり、帯域制御、遅延、パケットロスなどを制御可能となっている。

3.2 設計方針と実装

jail を操作するために従来はシェル上で CUI を用いていた。そして VIMAGE を用いて複雑なネットワークを構成するためには延々とコマンドを打つか、次のようなシェルスクリプトを組む必要があった。

```
for group in ‘seq 0 2‘
  do
    epairnum=’expr $epairnum + 1’
    t1=’expr $group "*" 3 + 1’
    ifconfig epair${epairnum}a vnet router0
    ifconfig epair${epairnum}b vnet router${t1}
    jexec router0 ifconfig bridge1 addm epair${
        epairnum}a
    jexec router0 ifconfig epair${epairnum}a up
    jexec router${t1} ifconfig epair${epairnum}b up
    epairnum=’expr $epairnum + 1’
    t2=’expr $group "*" 3 + 2’
    ifconfig epair${epairnum}a vnet router${t1}
    (snip)
done
```

もう少し直感的にわかりやすいプログラミングをしたいと考えた結果、Ruby の Shell クラスでラッパーを書くことを思いついた。Shell クラスではコマンドを関数化することができる。

次のようにコマンドを定義しておけば、

```
Shell.def_system_command(’ifconfig_org’, path = ’/
    sbin/ifconfig’)
def ifconfig(param)
  sh=Shell.new
  sh.transact{
    ifconfig_org(*param.split)
  }
end
```

このように使うことができる。

```
ifconfig(’epair0a inet 192.168.1.1 netmask
    255.255.255.0’)
ifconfig(’epair0a up’)
```

また操作対象をオブジェクト化することで操作対象に命令 (メソッド) を送るという直感的なプログラミングが可能となる。

次のようにルータクラスを定義しておけば、

```
class Router
  def initialize(jailname)
    ...
    jail(jailname)
    jexec(jailname,’ifconfig lo0 127.0.0.1/24 up’)
  end

  def assignip(epair,ip,mask)
    ...
  end
end
```

ルータへのアドレス設定の操作は次のようになる。’router0’ という名の jail を用いて router0 という名のルータオブジェクトを作成し、assignip メソッドで IP アドレス設定が行える。

```
router0=Router.new('router0')
router0.assignip('epair0b
    ','192.168.1.254','255.255.255.0')
```

ここで困ったことに直面する。たくさんのオブジェクトを作って操作したいときにオブジェクト名をどうするのかという問題である。Ruby では以下のような感じでオブジェクト名に変数を用いることはできない。(メタプログラミングを除く)

```
0.upto(10) do |num|
  "router#{num}"=Router.new()
done
```

そこで次のようにリフレクション (eval) を用いることにした。

```
0.upto(10) do |num|
 eval("router#{num}=Router.new('router#{num}')")
done
```

しかしエンドユーザに eval を多用させるのは面倒だし好ましくない。メタプログラミングも考えたがライブラリが難読化しそうであるし、もっと単純な手はないかと考えて生まれたアイデアが、機器を操作するオペレータ (ネットワーク運用技術者) を抽象化し擬人化したクラスを用意して eval を隠蔽してしまうという手であった。

```
class Operator
  def setupserver(jailname)
    eval("$#{jailname}=Server.new('#{jailname}')"
        )
  end

  def createpair
    ...
  end

  ...
```

こうしてオペレータクラスからオペレータオブジェクトを生みだせる。

```
tomocha=Operator.new
```

このオペレータオブジェクト tomocha さんにはルータをセットアップしてもらうことができる。次の例では route1, router2 という jail を起動しルータオブジェクトとして扱えるようになる。

```
tomocha.setuprouter(router1)
tomocha.setuprouter(router2)
```

tomocha さんにルータとルータをつないで頂くこともできる。createpair は jail における仮想 LAN ケーブルを生成し、connect でその両端を機器オブジェクトに接続する。

```
a,b=tomocha.createpair
tomocha.connect(router1,a)
tomocha.connect(router2,b)
```

このようにして生まれたクラスライブラリ VITOCHA [8] によって、プログラマは自在に仮想ネットワークを構築することができるようになった。また構築したネットワークの接続関係をグラフ化し、FLOSS の nwdiag [9] によってネットワーク図を生成する付加機能も用意してある。

3.3 ライブラリ構成

VITOCHA の構成は以下のようになっている。

vitocha.rb メインライブラリ
shcommand.rb jail のコマンド群を定義するクラス
equipment.rb 機器クラス
bridge.rb スイッチクラス (機器クラスのサブクラス)
router.rb ルータクラス (機器クラスのサブクラス)
server.rb サーバクラス (機器クラスのサブクラス)
mkrouter ルータ用の jail ファイルシステム生成スクリプト
mkserver サーバ用の jail ファイルシステム生成スクリプト
unjail.rb jail 破棄用スクリプト

ソースコードは BSD ライセンスで次章に紹介するサイトで仮想アプライアンスに包含して公開している。

4. インターネットシミュレータ II

VITOCHA を使って筆者がまず作成したのは類似システムの章で紹介した初代インターネットシミュレータをコンパクトに再構成したインターネットシミュレータ II である。

インターネットシミュレータ II では 4 つの AS(仮想 ISP) が BGP4 で繋がり、それぞれの AS 配下では OSPF と RIP が動いている。10 台のルータ、4 つのブリッジ、4 つのサーバから構成され、DNS ルートサーバのもとに .nom ドメインが運用されている。**図 1** は VITOCHA のネットワーク図生成機能が出力したものである。

このシステムに GUI をつけたものをウェブサイト [10] で公開しており、ルータのコンフィグ、ルーティングテーブル、DNS 検索や traceroute の結果などを見ることができる。また、環境をまるごと VirtualBox で実行できる仮想アプライアンスとして公開しており、ダウンロードして手元で利用することができる。

5. 研究への適用

筆者の最近の研究の主テーマである DNS の脆弱性の研究に VITOCHA は非常に役立っている。BIND や NSD,Unbound 等の各種バージョンのサーバソフトウェア実装を仮想環境にインストールし、ルートサーバからの階層的なドメイン名空間を構成し、各種実験を行っている。特にキャッシュポイズニングの実験においては DUMMYNET の機能で自在に遅延を入れられることが実験の効率を高め

てくれた。これらについては別論文を色々と発表したいところであるが、解説に賛否のあるセキュリティ情報が多く、あまり論文化できていない。e-ontap [11] を中心にネット上でいろいろ情報発信を行っているので参照されたい。

6. 教育への適用

VITOCHA は教育においてとても役立っている。 VITOCHA はプラットホームの OS が普及度の低い FreeBSD であるが、OS ごと環境を作り上げた OVA フォーマットの仮想アプライアンスとして配布することにより、FLOSS の

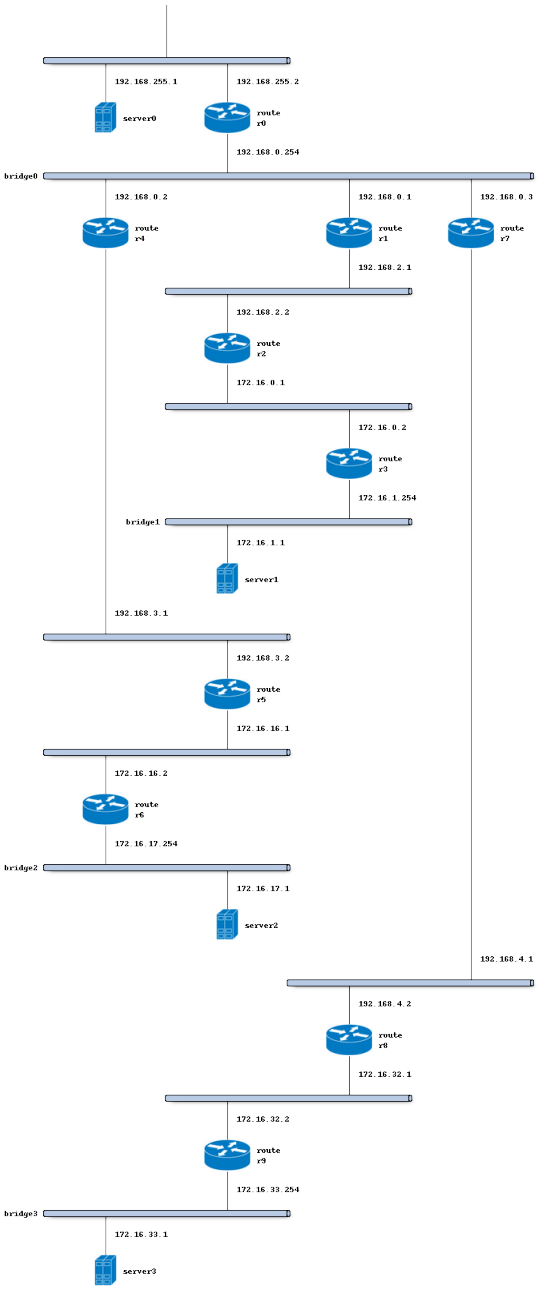

図 1　インターネットシミュレータ II

VirtualBox をインストールしたパソコンであれば、Windows や Mac その他各種 OS の上の VM で動作させることができる。以下にこれまで VITOCHA を用いて行ってきた技術者教育について紹介する。

6.1 DNS 温泉

2014 年から毎年 DNS 温泉という合宿形式の DNS 勉強会を開催している。これまで第 1 回の猿投温泉に 18 人、第 2 回の熱海温泉に 32 人、第 3 回の玉名温泉に 14 人、第 4 回の山城温泉に 30 名の技術者たちが集まった。深く議論したいため人数は制限をしている。また 2016 年に東京で行った番外編には 58 人が参加した。

DNS 温泉では主として筆者が講師となって DNS の仕組みや諸問題についての講義の他、VITOCHA を用いて用意された演習用仮想アプライアンスを各自が持ち込んだノートパソコンに読み込み、各種の演習を行っている。演習環境にはルートサーバの他、数個のトップレベルドメインと 10 数個のセカンドレベルドメインが用意されており、ゾーンの分割と委譲による DNS の階層構造が学べる他、ゾーン構成やキャッシュサーバの実装の違いによる応答の違いや脆弱性を実験によって確認するカリキュラムを組んでいる。テキストはウェブサイトで公開 [12], [13], [14], [15] しているので参照されたい。

6.2 v6tokai

産学連携 IPv6 実験地域ネットワーク研究会 (v6tokai) を 2010 年から筆者が主宰となり続けてきている。その中で VITOCHA を用いて、IPv6 の ND プロトコルや RIPng,BGP4+などのルーティングを実践的に学ぶハンズオンを時々行っている。

6.3 情報工学実験

筆者の中京大学には情報工学実験 I という科目がある。5 人の教員が 5 つのテーマの実験を受け持っており、筆者は 3 週計 12 時間のネットワーク実験を担当している。この実験では TCP の特性である帯域遅延積を理解する課題を当面扱っている。選択科目で受講者は 30-40 人程度である。

ネットワーク実験において学生は仮想アプライアンスが入った UCB メモリを受け取り、演習室のパソコンにインストールされている VirtualBox で読み込み環境を起動する。

Unix 系 OS やエディタの使い方もままならず Ruby のプログラミングも始めての学生たちが、一週目には授業の説明の時間を除いた 2 時間程度で 2 つの仮想マシンと仮想ブリッジを起動し接続するスクリプト作成と操作を行う。出来上がった**図 2** のネットワーク上で ping を通し、自分で設定した FTP サーバから dd で作成したデータファイルをダウンロードできれば実験環境の完成である。

演習室では困惑から始まり、TA や教員にヘルプを求め

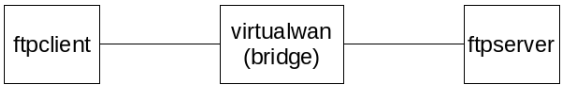

図 2　情報工学実験ネットワーク

る挙手の山、ping が通った時の歓喜の声まで熱い戦いが繰り広げられる。

2〜3 週目には 2 つの仮想マシンの間を繋ぐ仮想ブリッジの DUMMYNET を操作し、帯域、遅延、パケットロスを変化させ、FTP のスループットを計測するのが実験の課題となる。実験を終えた学生たちはその結果をレポートにまとめて提出する。**図 3** に学生がレポートに載せる実験データをまとめたグラフの例を示す。

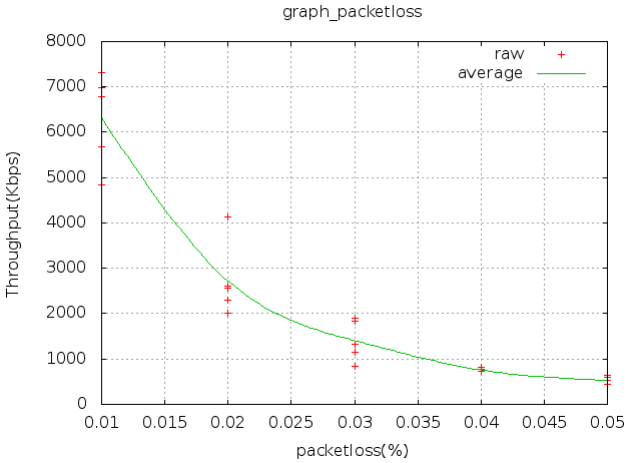

図 3　実験データのグラフ例

7. まとめ

仮想ネットワーク構築ライブラリ VITOCHA とその教育への適用例について報告した。ノートパソコン上で手軽にネットワークやサーバを構築して実験が行えるため応用範囲は広いだろう。ライブラリ自体は研究というより実用優先で作っており、特段の性能評価などは行っていないが主記憶 8GB 程度のミドルクラスマシン上でも数十のルータやサーバは動かせるはずだ。筆者の研究室でもゼミ生の教育や卒研のプラットホームとして活用が進んでいる。今後はゼミや情報工学実験 I だけでなく、コンピュータネットワークやセキュリティ理論の授業等でも活用していきたいと考えている。また一般のネットワーク技術者の教育への展開も検討しており、協力者、協力組織を募っていきたい。

参考文献

[1] 鈴木常彦: インターネットシミュレータの構築報告, 日本ソフトウェア科学会, 2008, http://wit.jssst.or.jp/2008/final-data/08006.pdf
[2] University of Zagreb: Integrated Multiprotocol Network Emulator/Simulator, http://imunes.net/
[3] M. Zec: Implementing a Clonable Network Stack in the FreeBSD Kernel,Proceedings of the 2003,San Antonio, Texas,USENIX Annual Technical Conference,2003.
[4] nsnam.org: ns-3,https://www.nsnam.org/
[5] CISCO: CISCO Network Academy, https://www.netacad.com/
[6] CISCO: PacketTracer, https://www.netacad.com/courses/packet-tracer
[7] 国立研究開発法人情報通信研究機構: StarBED4 プロジェクト, http://starbed.nict.go.jp/
[8] 鈴木常彦: VIMAGE 仮想ネットワーク構築クラスライブラリ VITOCHA, 2012, http://www.e-ontap.com/misc/virtualtomocha/
[9] blockdiag:ネットワーク図生成ツール nwdiag, http://blockdiag.com/ja/nwdiag/index.html
[10] 鈴木常彦: Internet Simulator II, 2012, http://sim.internot.jp/
[11] 鈴木常彦: インターネット崩壊について考えるページ, http://www.e-ontap.com/internet/
[12] 鈴木常彦: DNS 温泉 2 テキスト, 2015, http://www.e-ontap.com/dns/onsen2/
[13] 鈴木常彦: DNS 温泉 3 テキスト, 2016, http://www.e-ontap.com/dns/onsen3/
[14] 鈴木常彦: DNS 温泉番外編テキスト, 2016, http://www.e-ontap.com/dns/onsenextra2016/
[15] 鈴木常彦: DNS 温泉 4 テキスト, 2017, http://www.e-ontap.com/dns/onsen4/
[16] 産学連携 IPv6 実験地域ネットワーク研究会 (v6tokai): http://www.e-ontap.com/convivial/society/