# 大規模ネットワークのエミュレーションと評価

2005MT019 後藤 聡　　2005MT081 小畑 隆治
指導教員　後藤 邦夫

## 1　はじめに

本研究の目的は広域ネットワーク上に配置されたアプリケーションの性能評価のために 1 台の PC 上で複数のネットワークホストの構築を実現することである．

1 台の PC 上で複数のネットワークホストを構築するためには，ネットワークスタックの仮想化が必要である．2007 年度浅野，加藤の卒業研究「性能評価のためのホストエミュレーションの提案と評価」[4] では仮想ソフトウェアを用いて実現していた．仮想ソフトウェアとは複数のオペレーティングシステム (以下，OS) を一台の PC 上で同時に動作させるソフトウェアである．二人の研究はこの仮想ソフトウェアを使用することにより，ネットワークスタックを含めたホスト全体をエミュレートしている．これによってネットワークモデルを実ネットワーク上で実装する前に仮想空間で使用することが可能になる．

ホスト全体のエミュレートはネットワークスタックのみの仮想化と比べ，ホスト毎に異なる環境を設定することが可能であるが，メモリを非常に多く必要とするため，メモリ不足に陥りやすいなどの短所が考えられる．

近年，比較的容易に仮想ネットワークスタックのみの仮想化が使用できるようになった．そこで，本研究はネットワークスタックのみの仮想化を行い，1 台の PC 上で複数のネットワークホストを構築する．そうすることで削減できたメモリをスレッド起動に充てることにより，大規模なネットワークのエミュレートを目標とした．

後藤はシステムの理論，ネットワークモデルのシステム構築を担当し，小畑はシステムの環境構築，プログラムの作成を担当する．

## 2　システム概要

本節では先行研究と仮想ネットワークスタックの概要，スレッドの概要について述べる．

### 2.1　先行研究

先行研究 [4] では仮想ソフトウェアを用いてホスト全体の仮想化をして複数のホストのエミュレートを行った．複数ある仮想ソフトウェアの中より性能評価をし，その結果「QEMU」と「User Mode Linux(以下，UML)」を用いて情報の使用量のシステムエミュレーションをした．家庭などにある装置のデータを ISP に置かれたサーバを介して，企業にあるセンターサーバへ送信するシステムを図 1 に示す．実ホストは 32bit 環境の PC を 2 台使用．センターサーバと各家庭にある装置をゲスト OS として，ISP サーバを別ホストとしてエミュレートすることによってシステム構築をし装置数を多くするために，プログラムによって 1 仮想ホスト内で複数のスレッドを使用している． 実験結果 QEMU では 3198

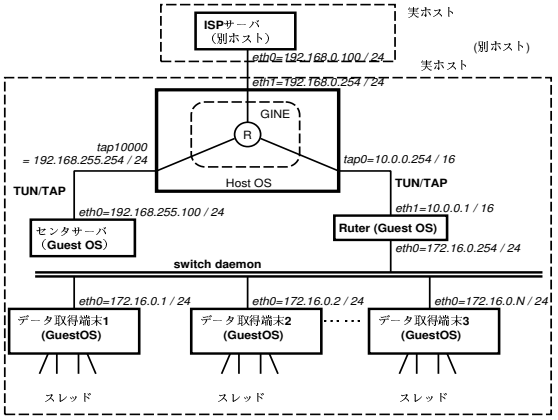


図 1　先行研究で提案されたモデル図 (UML 使用)

個、UML では 6193 個の端末との通信に成功した．ネットワークスタックのみの仮想化を行うことで削減できたメモリをスレッド起動に充てることで，より大規模なネットワークのエミュレートが可能になると考えた．本研究はネットワークスタックのみの仮想化を用い，スレッドを多く起動することのできる 64bit 環境で実験を行う．

### 2.2　ネットワークスタックの仮想化

ネットワークスタックとは、ネットワークの構造を抽象化した OSI 参照モデルのようにネットワーク階層がつまれたものをいう．1 台の PC 内で複数のネットワークホストを構築するためには，TCP や UDP でのポート番号を変えることで最大ポート番号分のホストをエミュレートすることができる．しかしこの方法では，アプリケーション間の通信パラメータ設定や通信の独立性を保つことができない．そこで使用されるのが，ネットワークスタックの仮想化である．仮想ネットワークスタックとは複数のネットワークインタフェースを 1 台の PC 上で同時に動作させるカーネル固有のシステムコールである．OSI 参照モデルではデータリンク層の Device driver(ネットワークデバイス) からネットワーク層までを仮想化する [2]．図 2 に示す．

### 2.3　仮想ネットワークスタックの実現

仮想ネットワークスタックカーネルは，ネットワークスタックの仮想化を行うカーネルの機能をパッチであてた Linux-2.6.20-netns1 と標準で組み込まれている Linux-2.6.26 の 2 つのカーネルが存在する．以下にそれぞれの特徴を述べる．

- Linux-2.6.20-netns1

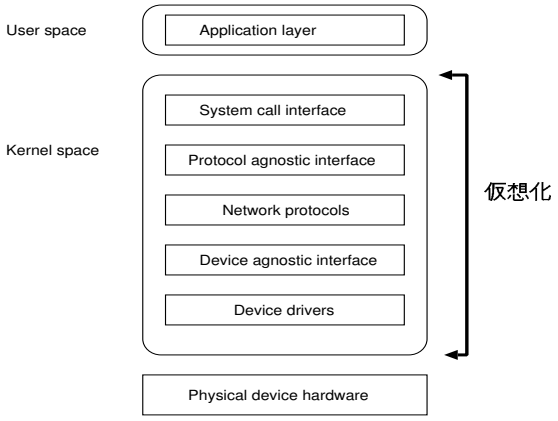


図 2　OSI 参照モデル

正式版ではないが，仮想ネットワークスタックである NETNS が比較的容易に設定できる．仮想ネットワークデバイスは，Ethernet tunnel device(etun) が使われる．

- Linux-2.6.26
  正式版ではあるが，NETNS を設定する際に SYSFS を使うことができなくなる．これによって，デバイス関係が使えなくなる．仮想ネットワークデバイスは Virtual ethernet pair device(VETH) が使われる．

その後カーネル独自のシステムコールを利用し，新しい net namespace を作る．仮想ネットワークデバイスでネットワークインタフェースペアを作り，片方を namespace に入れる．そうすることによって独立したネットワークインタフェースをもつ仮想ネットワークスタックを実現することができる．

### 2.4　64bitOS の特徴

1: 処理能力が格段に上昇し，短時間に大量のデータを処理できる．

2: 64bit CPU では扱えるメモリーも格段に増える．

3: pid_max の値を 2 の 22 乗まで変更できる．

一般的な 32bit CPU では、PC に搭載できるメモリーは最大 4GB（約 40 億バイト）までだが、64bit CPU なら理論上は最大で約 172 億 GB のメモリーを扱えることになるので大量のデータの処理など高速、効率的に実行する．

また，32 ビットのプラットフォームでは，pid_max(プロセスではないが) スレッド数も含むパラメータの最大値は 32768 であるの に対し 64 ビットのプラットフォームでは，2 の 22 乗 (PID_MAX_LIMIT, 約 4,000,000) までの任意の値を設定できる．

以上より，32bitOS ではなく 64bitOS を使用することで，最大スレッド数の増加と実行効率の向上が期待できると考えた．

## 3　仮想ネットワークスタックとスレッドの性能評価

本研究では PC の資源を増やさずにより多くのスレッドを起動することを目標としているためネットワークスタックの仮想化を用いてホストの仮想化をする．

実際にシステム性能評価を行う PC を用いてどれだけの仮想ネットワークスタックが起動するか，どれだけのスレッドが並行するかを調べる．

### 3.1　仮想ネットワークスタック実験

1 台の PC で仮想ネットワークスタックが並行する限界を調べるために実験をした．カーネル独自のシステムコールを呼び出し，動的に仮想ネットワークスタックを作るプログラムを使用した．メモリ量の計測には top コマンドを使用した．結果を表 1 に示す． 表 1 より，一つ

表 1　仮想ネットワークスタックの並行実験

| | メモリ | 使用メモリ | 使用 swap メモリ |
|---|---|---|---|
| 初期段階 | 2052332 | 368204 | 0 |
| 500 個目 | | 873812 | 0 |
| 1000 個目 | | 1731912 | 0 |
| 1200 個目 | | 2035560 | 1464 |
| 1500 個目 | | 1811924 | 925496 |
| 2000 個目 | | 1382628 | 2894296 |

(単位：kb)

の仮想ネットワークスタックの起動に約 1200kb 必要となる．また，仮想ネットワークスタックを 1200 個前後起動すると実メモリを使いきることが分かる．1200 個以降は swap メモリを使用し起動する．非常に時間がかかるため本研究では 2000 個までしか実験をしていないが，swap メモリがなくなるまで起動すると考えられる．

### 3.2　スレッド実験

1 台の PC でスレッドが並行する限界を調べるために実験を行った．pthread_create() でスレッドを起動し，nanosleep() で待機させることでスレッドを並行起動するプログラムを用いて実験した．

root ユーザでは/proc/sys/kernel/にあるスレッド並行数の最大値を設定している threads-max とプロセス番号の最大値を設定している pid_max を 32744 から 1000000 へ変更した．

pthread の実装について，Vine Linux(32bitOS) では LinuxThreads[1]，Ubuntu(64bitOS) では NPTL を使用している． 表 3 より最大スレッド並行数は 4 プロセ

表 2　Vine Linux 4.2(32bit) での実験結果

| | 一般ユーザ | root ユーザ |
|---|---|---|
| 1 プロセス目 | 16366 | 16381 |
| 2 プロセス目 | shell 起動 不可 | 16032 |
| 3 プロセス目 | | shell 起動 不可 |
| 合計並行数 | 16366 | 32413 |

(単位：個)

表 3　Ubuntu 8.04(64bit) での実験結果

| | 一般ユーザ | root ユーザ |
|---|---|---|
| 1 プロセス目 | 16325 | 32757 |
| 2 プロセス目 | shell 起動 不可 | 32757 |
| 3 プロセス目 | | 32757 |
| 4 プロセス目 | | 20652(メモリ不足で停止) |
| 合計並行数 | 16325 | 118923 |

(単位：個)

ス，118923 個である．

実験結果より，一般ユーザと root ユーザではアクセス権限の違いによりスレッド並行数に大きく差があることが分かる．また，同じメモリ量であっても，64bit 対応のカーネルの方が多くのスレッドを起動することができる．1 スレッドあたり 16KB 程度で 1 プロセスあたり最大で約 520MB 必要であることが分かる．また，スレッドでは swap メモリは使用できない．

より大規模なネットワークをエミュレートするには 64bit 対応カーネルを導入できる Linux ディストリビューションで行うと良いことが分かった．

## 4　自動データ取得システム

本節では，仮想ネットワークスタックを用いたシステムのエミュレーションと評価を行う．

### 4.1　システム仕様

- ある会社のサービス世帯数より，端末数が合計 1000 万，ISP サーバが 50 箇所と想定する．よって 1 台の ISP サーバで受け持つ端末数は 20 万台となる．
- データ取得は端末と直接通信を行うサブサーバとサブサーバと通信をする ISP サーバの 2 段階で行われる．
- 各端末は登録された端末 ID, ISP サーバと共有する秘密鍵，自分の秘密鍵を持つ．付録にあるように XOR(排他的論理和) により双方を認証する．端末，ISP サーバともに 8bits で暗号強度 16bits となる．
- GINE を用いて ISP サーバと各端末間で遅延，パケットロスを起こすことで実ネットワークに近い模倣をする．
- サブサーバと各端末はそれぞれ固有の port 番号を持っている (IP アドレスは同じ)．
- サブサーバと各端末間の通信は非常に低速 (約 5kbps) である．
- 図 3 にシステムのモデル図を示す．

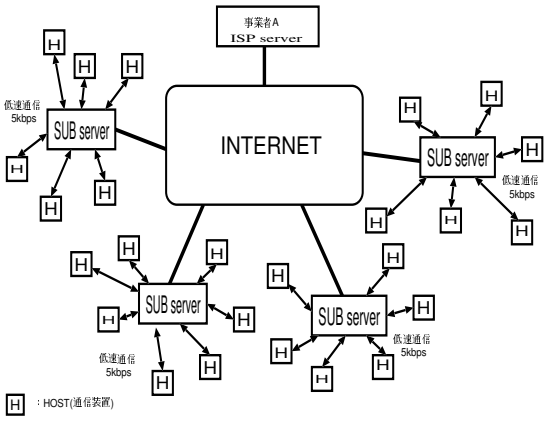


図 3　自動データ取得システムのモデル図

### 4.2　システムの実現

前述の 4.1 のシステムをエミュレートする方法を提案する．

- ISP サーバを別ホストとしてエミュレートし，サブサーバは仮想ネットワークスタックを用いて仮想ホストとしてエミュレートした．仮想ホスト上で起動したスレッドを端末とすることでシステムを構築する．
- 一度に目標とする 20 万個のスレッドの起動はメモリの制約上できないので，1 つのサブサーバを 1 つのグループとして稼働端末のみエミュレートするように構成した．
- 1 つのグループとの通信は 300 秒とし、この時間内に通信できなかったものは時間切れのエラー処理を行う．
- IPv6 を用いて通信を行った．
- サーバと端末の間には 1 つルータを用意し，アドレス変換を行った．
- サブサーバと各端末間の通信は GINE を用いて端末の通信のみ様々な帯域を設定した．
- 図 4 に仮想ネットワークスタックを用いたモデル図を示す．

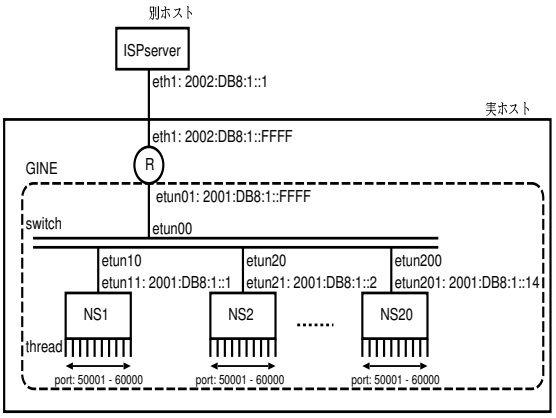


図 4　自動データ取得システムのモデル

#### 4.2.1　GINE の概要

本研究でエミュレートするシステムには NFQUEUE を用いた GINE[3] を使用した．

- NFQUEUE とはパケットをユーザ空間のプログラムやアプリケーションへキューイングするカーネルの機能である．
- NFQUEUE を組み込むために, ユーザプロセス側で libnfnetlink-0.0.33, libnetfilter_queue-0.0.15 の 2 つのライブラリをインストールする．
- ip6tables コマンドを用いて NFQUEUE ターゲットを指定して queue number(unsigned integer, 0 - 65535) を指定する．
- 指定された queue number を横取りし、処理を行う Queue へ流し、元の経路へ戻す．
- /sbin/sysctl -w net.ipv6.conf.all.forwarding=1 を実行する必要がある．
- 端末とサーバ間の通信は 5kbps * 端末数 以下に様々な帯域設定を行ってエミュレートした．

#### 4.2.2 GINE 内の構成

GINE 内の NFQUEUE の構成を図 5 に示す． 流れて

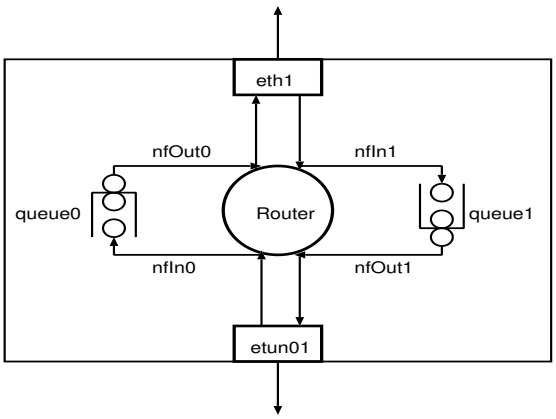


図 5 GINE を用いたネットワーク構成

きたパケットは GINE 内で作成した NFQueueIN クラスで横取りをして FrameQueue に入れ，NFQueueOUT クラスで出力している．

### 4.3 エミュレーション結果の評価

表 4 にエミュレーション結果を示す．

表 4 エミュレーション結果

| 通信帯域 | 成功したスレッド数 (個) |
|---|---|
| 1000Mbps | 196401 |
| 50Mbps | 173720 |
| 10Mbps | 171013 |
| 5Mbps | 165680 |
| 1Mbps | 98136 |
| 500kbps | 106766 |
| 100kbps | 3522 |
| 50kbps | 1319 |
| 10kbps | 68 |
| 5kbps | 70 |

- 端末スレッドは 10000 個起動しているが，通信の失敗によりそれより少数のデータしか取得できなかった．
- 帯域設定を小さくすると処理能力以上のパケットが流れ、通信可能端末数が減少する．
- 通信のプロトコルに UDP を使用しており，同時に大量のパケットを送信すると多大なパケットロスが発生する．そのための再送の処理がプログラム上で行えていないために多くのエラーが発生してしまった．

## 5 おわりに

本研究を通して, 仮想ネットワークスタックを使用し, 仮想ネットワーク上に配置されたアプリケーションの性能評価を行った結果，先行研究のシステムでは ISP サーバが一度に 1500 個の装置に対してしか動作しないという課題を改善することができ，目標としていた 20 万個の端末との通信に成功した．しかしこの結果はサーバと端末の間のネットワークの通信速度を制限していないものである．本研究ではサーバと端末間では無線を想定した低速通信 (5kbps) での実験を行っていたが，その実験では端末との通信に幾らかの失敗が起こった．以下に今後の課題を述べる．

- より大規模でのエミュレートを想定する場合，本研究で使用した PC を複数台用意したり，搭載するメモリ量を増やすことで，ゲスト OS の使用メモリ量を増やすことで，プログラムによって起動するスレッド数を大幅に増加することができると考えられる．
- サーバと端末間での低速通信の実現．
- 本研究のエミュレート方法では 1 つの NS を 1 つのグループとして unicast を使用し，1 万個ずつエミュレートしたが, 1 つのグループで複数の NS を起動し，multicast を使用することで，数万個ずつエミュレートすることが可能と考えられる．

## 参考文献

[1] *The Linux Threads Library* (accessed Jul. 2008). http://pauillac.inria.fr/~xleroy/linuxthreads/.

[2] Hunt, C.: *TCP/IP*, Vol. 2, chapter 1, pp. 7–13 (2001).

[3] Sugiyama, Y. and Goto, K. (Eds. Zhang, S. e. a.: Design and Implementation of a Network Emulator using Virtual Network Stack, *Proc. of the Seventh International Symposium on Operations Research and Its Applications (ISORA2008)*, World Publishing Corporaiton, pp. 351–358 (2008).

[4] 浅野洋介, 加藤史章：性能評価のためのホストエミュレーションの提案と評価，卒業論文，南山大学 数理情報学部 情報通信学科 (2007).