大規模ネットワーク構築のための GINE の管理機能の追加

M2008MM002 浅野洋介
指導教員：河野浩之

1 はじめに

Web サービスやインターネットを利用したアプリケーションはさまざまなネットワーク障害を想定して検証する必要がある．そこで使われるのがネットワークエミュレータであり，これを用いてホスト間の経路を模倣することで，さまざまな状況下での実験が可能である．

本研究の目的は，現実的なネットワークにより近い環境を模倣するために，後藤研究室で開発中のネットワークエミュレータ **Goto's IP Network Emulator**(以下，**GINE**) を改良すること，そして新機能を追加することである．2008 年の研究 [6] では仮想ネットワークスタック機能を組み込むことに成功し，実ネットワークに近い環境を模倣できるようになった．しかし，これによって複雑なネットワークを構築できるようになった一方で，システムを管理する機能が必要になってきた．

本研究では，GINE の操作性向上と大規模ネットワークの構築を想定し，GUI を用いたエミュレーションモデル等の保存/読出機能や実行時の操作機能などの管理機能を追加する．

2 GINE の概要

GINE とは多数のルータやリンクで構成されるネットワークを模倣できるネットワークエミュレータである．これは IPv4/v6 のリンクエミュレーションや IPv4/v6 アドレス，プロトコル，ポートのフィルタリング，さらにルータ/ホストのエミュレーションも可能である．GINE でのエミュレーションはカーネルレベルではなく，ユーザプロセスレベルで実現されている．そしてシステムコールを多用するため C++を，マルチスレッドを使用するために **GNU common C++**[1] の *Thread* クラスを用いたプログラムで記述されている．

2.1 リンクエミュレーション

GINE で提供するリンクはプログラムによって生成される独自 Queue で表現している．この Queue は有限長の双方向リストであり，各リンク毎に与えられた確率分布に従ってフレーム遅延/損失などの通信障害や指定した帯域幅でリンクを模倣することができる．フレームが Queue に到着したとき，到着時間に一定またはランダム遅延時間を追加し，出発時間順で並び替えることで出発時間でフレームが出力される．

外部ホストからフレーム (パケット) をエミュレータ内に注入するためには，横取り機能が必要である．GINE では，Linux カーネル 2.6.14 から標準搭載されている Netfilter 機能の **NFQUEUE** を用いる．*iptables* のフィルタリングルールをプロトコル，経路など異なった種類や，往き復りのリンクごとに設定し，それぞれ別の NFQUEUE を設定することで違うパラメータの障害を発生するとが可能である．

2.2 ホスト/ルータエミュレーション

GINE 内で仮想的にホストやルータ等を模倣するためには，プログラムでそれらの機能を実装するか，ネットワークスタックの仮想化を利用する必要がある．仮想ネットワークスタックとはルーティングやフォワーディング機能を含めたネットワーク部分を仮想化し，ネットワークホストを構築する方法である．[6] によってネットワークスタックの仮想化技術である **Network Namespace**[4](以下，**NETNS**) を GINE に導入し，実ホスト，実ルータ等の機能をより容易に実装できるようになった．これは，Linux カーネル 2.6.26 から標準で搭載されており，カーネル修正は不要である．

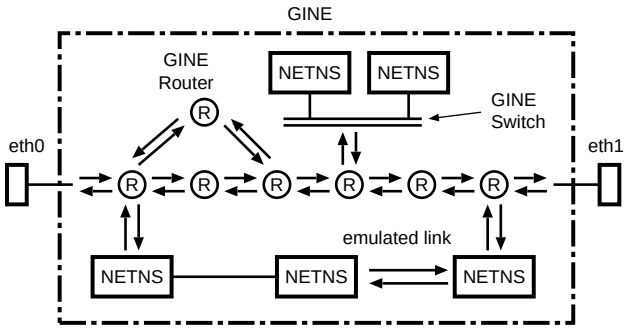

図 1 ホスト/ルータエミュレーション

図 1 のように仮想ルータ，仮想スイッチングハブなどをプログラムによって生成し，さまざまなエミュレーションモデルを構成することができる．仮想ネットワークスタックによって生成されたネットワークホストを用いた場合，1 つの OS 内で複数のネットワーク環境を利用できるため，ホストごとに独立したネットワークインタフェースやルーティングテーブルを使用することができる．また NETNS にルータ機能を持たせたり，RIP や BGP などのルーティングデーモンを起動することで，GINE Router を用いるより，容易で現実的なネットワークを構築できる．なお図 1 ではホスト 1 台のみ使用しているが，外部ホストと接続しフレーム横取り機能を利用することで実ホスト間でのエミュレーションも可能である．

2.3 改良点の概要

本研究では，大規模ネットワークの構築を想定して，機能を充実し，かつより容易に GINE を使用できるようにするために，GUI ツールを作成する．またそれに必要な機能として「NETNS の端末管理機能」,「ネットワークの保存/読出機能」を実現する．なお，この 2 点においては CUI でも動作できるように実現する．

検証に使用したホストは DELL PowerEdge 840 (Intel Xeon X3220 2.40GHz，Core 2 Quad (core 4))，Ubuntu 9.04 64bit OS である．

3　管理機能の追加

GINE に搭載する管理機能について説明する．

3.1　仮想ネットワークスタックの自動生成と端末管理

現在動作しているプロセス上で，*syscall* 関数と *execl* 関数を用いて bash シェルを起動することによって新しいプロセスへ置き換わり，NETNS となる．これらの関数を含めたプログラムを実行した端末で NETNS を操作できる．つまり仮想ネットワーク数が大きいとき，同数の端末を起動することにより無駄なメモリ消費が増え，ユーザの操作も困難になる．GINE では NETNS を複数起動させることを想定しているため，この方法では無駄なメモリを消費する可能性がある．[6] では GINE クラスで NETNS の処理を管理し，バックグラウンドで動作する機能を定義している．しかしこの方法では端末によるマニュアル (対話型) 操作ができなくなっていた．

本研究ではバックグラウンドにある NETNS を操作する機能の改良に加え，NETNS をマニュアルで操作するための端末を開閉する機能を実現した．端末には `xterm` を使用する．図 2 にその概要を示す．

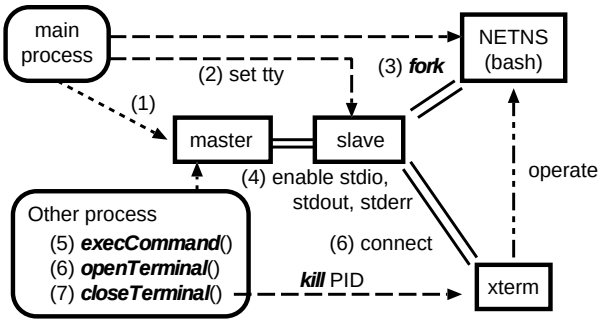

図 2　NETNS 生成と端末開閉機能

(1) キーボード以外のプロセスからのアクセスを有効にするために，`/dev/ptmx` からマスタとスレーブの疑似端末のペアを生成．

(2) `/dev/tty` から制御端末の端末情報を取得し，スレーブ端末にセット．

(3) *fork* 関数でプロセスを分岐し NETNS 生成，シェル (bash) をセット．

(4) スレーブ端末に対して標準入出力を可能にする．

(5) 他プロセスからマスタ，スレーブを経由してコマンドを実行．

(6) マスタ，スレーブの tty を指定して `xterm` を起動．

(7) `kill xterm_PID` で `xterm` を終了．

改良によってバックグラウンドの NETNS へコマンドを送信し，実行する機能と端末の開閉機能を実現できた．

3.2　オブジェクトの保存/読出機能の追加

さまざまなアプリケーションには任意のデータをファイルに保存したり，読み出したりする機能がある．特に動的に変化するモデルや複数の型を持つオブジェクトなどを生成するためには，オブジェクトの状態等を保存し復元する機能が必要である．しかし現在の GINE にはそのような機能は存在しない．

本研究では *Thread* クラスを利用している GNU common C++にある Persistence[1] を用いて，オブジェクト自体とその識別子をファイル化する機構 (serialization，直列化) を実現する．使用した Persistence のクラス/メソッドについて説明する．

- *Engine* クラス
  - データを保存する/読み出すファイル名を指定した C++ STL(Standerd Template Library) の *fstream* 型の変数を指定し，ファイルモードを変更してファイルへ書き込む/ ファイルから読み出すことができる．
- *BaseObject* クラス
  - *Persistence* オブジェクトを生成するためにクラスの型を定義するクラスであり，オブジェクトを保存したいクラスでこのクラスを継承することで Persistence を実現することができる．
  - データ構造は自動的にシステム内で管理される．
  - *write*(*Engine*& `archive`) メソッド
    クラス内で定義した変数，オブジェクト (OS のソケット，実行したコマンド類) をオペレータ”<<”を使用して保存するメソッドである．
  - *read*(*Engine*& `archive`) メソッド
    write メソッドで保存した変数，オブジェクトをオペレータ”>>”を使用して読み出すメソッドである．なお，読み出しの順番は *write* メソッドで保存した順番と一致していなければならない．
  - *getPersistenceID*() メソッド
    オブジェクトの識別子 (Class ID) を取得するメソッドである．

GINE で生成したオブジェクトはほとんどが双方向リスト形式で表現されているため，先頭のオブジェクトを指定して保存すれば末尾オブジェクトまで保存することができる．

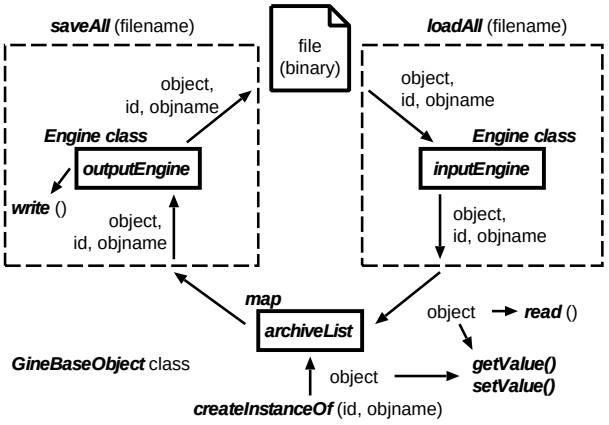

図 3　オブジェクト保存/読出

オブジェクトを保存/読出する流れを図 3 に示す．GINE にこれらの機能を組み込むために，*BaseObject* クラスを継承する *GineBaseObject* クラスを定義し，このクラス

を GINE クラスで継承した．このクラスは各クラスのオブジェクトを生成するメソッドやオブジェクトをファイルに保存/読出するメソッドを定義した．また各クラスの共通処理をこのクラスにまとめ，各クラスから継承する方法に変更した．

1. プログラム内では C++ STL の *map* コンテナで作成したこのクラスの静的メンバ変数 (`archiveList`) にオブジェクト名を要素としてオブジェクトを格納する (*createInstanceOf* メソッド)．このオブジェクトに必要な情報は *setValue* メソッドで定義する．
2. 1. で格納したオブジェクトを *Engine* クラスを用いてファイルに出力する (*saveAll* メソッド)．
3. ファイルから *Engine* クラスのオブジェクトに格納し，オブジェクトを読み出す．このとき *read* メソッドが呼ばれ，固有情報を読み出し，*setValue* メソッドで定義する．

GineBaseObject クラスの実装と各クラスの仕様変更によりオブジェクトをファイルに書き出し，そのファイルからオブジェクトを読み出すことができるようになった．

3.3 GUI 機能の追加

現行の GINE はエミュレーションモデルごとにプログラムを作成し，コマンドベースで実行している．しかしこの方法ではプログラムを書くことができないユーザには使用することが困難である．

本研究では GINE ライブラリを操作する GUI ツールを作成し，ユーザに対してエミュレーションモデルの作成と実行を助ける．ツールにはさまざまなソフトウェアでの利用実績があり，オープンソース (C++) で配布されている **Qt**[5] を使用した．GINE ライブラリを GUI ツールから操作するために，3.2 節の *GineBaseObject* クラスをインタフェースとして実装し，GINE クラス内に Qt のプログラムを混在させないよう工夫した．最終的に図 4 のような GUI ツールを実現した．このツールの主な機能は次の 6 つである．

1. GINE クラスで生成するオブジェクトをボタンとして配置する．テキストファイルに 1 行ずつ記述した GINE クラスを読み出し，クラス名をボタンのラベルとして配置した．このテキストファイルに新たに GINE クラスを記述することで，容易にボタンを作成することができる．
2. オブジェクトを生成するボタンを描画フィールドにドラッグ&ドロップで配置する．ドロップした位置情報を格納し，GINE のオブジェクトを作成する．オブジェクトの生成には 3.2 節の *createInstanceOf* メソッドを使用した．オブジェクトを生成したときに，管理しやすいようにクラス名の先頭に整理番号を付与し，オブジェクト名とラベル名とした．
3. 描画フィールドにドロップしたラベルの右クリック時にコンテキストメニューを表示させ，オブジェクトの削除や NETNS ラベルでは 3.1 節の端末開閉機能を呼び出すメニューを追加した．
4. オブジェクトのボタンをダブルクリックすると別ダイアログが開き，各クラスの固有情報を設定する．例えば，FrameQueue クラスでは遅延，損失などの値を入力，どのオブジェクトに接続するのか，などを設定する．各クラスの *getValue* メソッドから固有情報を取得し，このダイアログに情報を設定する．情報を入力/削除/編集し，正しい値が入力されていれば，save ボタンをクリックすることで各クラスの *setValue* メソッドを呼び出し，情報をオブジェクトに登録する．
5. スタートボタンをクリックしたとき，スレッドがスタートして通信を開始する．またポーズボタンやストップボタンを用意し，時間経過に沿ったシナリオを実現する．スレッドスタートは *Thread* クラスの *start* メソッドを使用している．さらに 2008 年度までに GINE になかった機能として *Thread* クラスの *suspend* メソッド，*resume* メソッドを定義し，スレッドの一時停止/再開機能を実現した．
6. ファイルメニューからエミュレーションモデルを保存/読出する．3.2 節の *saveAll* メソッドを呼び出すことで GINE のオブジェクト関連は保存することができる．このときにドロップされたボタンとその位置情報も保存する．*loadAll* メソッドを呼び出すことで GINE のオブジェクト関連は読み出すことができる．そしてボタンを位置情報に基づいて配置する．

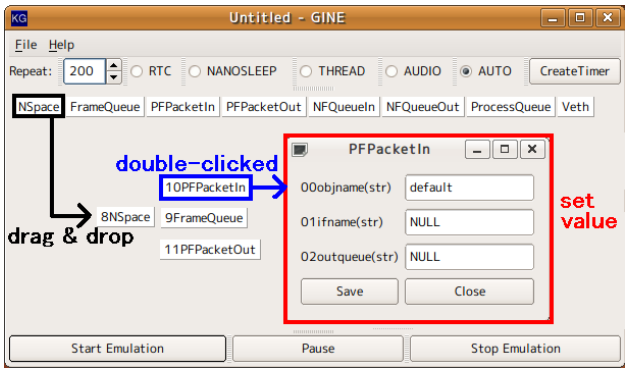

図 4 GUI アプリケーション

このツールによってプログラムを書くことができないユーザも GINE を容易に使用できるようになった．ウィンドウのサイズによるが，1 つの GUI ツールで構築できるノードは視認性の観点から 100 ノード前後である．

4 GINE の性能評価

3.3 節で作成した GUI ツールの評価と RIP を用いた小規模ネットワークを構築し，GINE の機能であるリンクのスループット性能を再検証する．

4.1 GUI ツール評価

本研究で作成した GUI ツールを利用者の視点から評価する．GINE の API を使用して CUI プログラムの記述経験がある学生 20 人に実際にツールを使用してもらい，アンケート 1(BAD)–5(GOOD) を実施した．その結果を表 1 に示す．

表 1 アンケート結果

	操作	手順	実行	保存/読出	総評
平均値	4.167	4.056	4.222	3.944	4.378

利用者からは，プロトタイプであったが全体的に使いやすく今後の展開が期待できるツールであるという高評価を得ることができた．しかし視覚的な補助機能がなく，作成できるモデルが限られているなど機能が少ないという指摘もあった．

4.2 ネットワーク評価

GINE の性能を再評価するために RIP を用いた小規模ネットワークを構築しスループットを測定した．図 5 のように NETNS を生成し，Virtual ethernet pair device(以下，veth) によって NETNS 間を直接接続する方法と veth と独自 Queue を用いて接続する方法の 2 通りで実験した．NETNS を仮想ルータ，仮想ホストとして配置し，IPv4 ネットワークを模倣する．この実験は Quagga[2] の RIPv2 を用いた．Quagga とは RIP，OSPF，BGP といったルーティングプロトコルをサポートし，オープンソースで公開されているソフトウェアである．NETNS 上でこのソフトウェアを起動させ，動的に経路制御する．

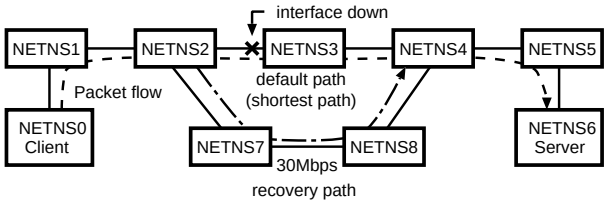

図 5 RIP ルーティングエミュレーション

1. スループット測定

   NETNS0 - NETNS6 間を独自 Queue あり/なしでスループットを測定した結果を表 2 に示す．結果，独自 Queue なしの場合，Gigabit インタフェース以上の高スループットを得ることが確認できた．独自 Queue ありの場合，複数用いることで，ホップ数に応じてスループットは徐々に減少する．これは GINE を実行しているホスト自体の処理性能の低下によるものと考えられるが，プログラムや処理方法で改善できる可能性がある．

表 2 スループット測定

from NETNS0	no Queue	with Queue
to NETNS1 (hop 1)	1542 / 1070	828 / 683
to NETNS2 (hop 2)	1013 / 1010	448 / 431
to NETNS3 (hop 3)	1010 / 907	302 / 240
to NETNS4 (hop 4)	925 / 889	234 / 180
to NETNS5 (hop 5)	861 / 818	194 / 162
to NETNS6 (hop 6)	800 / 766	176 / 104

(TCP / UDP，単位：Mbps)

2. インタフェース切断による通信経路の再計算

   RIP によって NETNS0–NETNS6 の最短経路で通信を確立後，NETNS3 のインタフェースをダウンさせ，RIP デーモンによる経路の再計算を確認した．結果，NETNS2 が NETNS3 への経路を削除し，NETNS3 へ最短で通信できる経路を知っている NETNS8 から新たな経路を受信するまでに合計約 150–180 秒必要で，その後，NETNS0 から NETNS6 へ再通信することが確認できた．またこのときのスループットは，GINE で設定した 30Mbps に近い，29.83Mbps の値を計測できた．

5 おわりに

本研究において GINE へ NETNS の端末管理，オブジェクト保存/読出機能を追加し，GUI ツールのプロトタイプを作成した．これらによって GINE の操作性が向上したと考える．

今後の課題として，単一または複数 PC 内での大規模ネットワークを構築するために GUI ツールのさらなる機能拡充が必要である．例えば，NETNS と GINE Router を併用し，RIP 等を用いて小規模ネットワークのグループを構築，保存する機能や，それらのグループ間を接続する機能が必要であると考える．そして複数の PC を接続し，その PC 間でネットワークインタフェースからのフレーム横取りと書き出しに NFQUEUE を用い，BGP 等の動的ルーティングデーモンを利用した大規模ネットワークを構築する機能が必要と考える．しかし大規模ネットワークを構築すると操作するホスト数が増加するため，経路制御が困難となる可能性がある．そこで [3] で提案された Quagga を操作する機能を GINE に組み込み，大規模ネットワークを構築できるように改良する必要があると考える．また GUI ツールは，視覚的な補助機能が少なく現状では複雑なネットワークを構築することはできないため GINE に存在する機能をさらに実装していく必要がある．さらにエミュレーションモデルのプログラムを生成する機能も必要であると考える．

参考文献

[1] Free Software Foundation：GNU common C++, http://www.gnu.org/software/commoncpp/ (accessed Apr. 2009).

[2] Ishiguro, K.：Quagga Software Routing Suite, http://www.quagga.net/ (accessed Apr. 2009).

[3] 星野聡介，石野佑弥：大規模ネットワークエミュレーションにおけるトポロジ構成と経路制御，卒業論文，南山大学 数理情報学部 情報通信学科 (2010).

[4] Network Namespace：Linux Containers, http://lxc.sourceforge.net/network.php (accessed Apr. 2009).

[5] Nokia：Qt - A cross-platform application and UI framework, http://qt.nokia.com/ (accessed Jul. 2009).

[6] Sugiyama, Y. and Goto, K.：Design and Implementation of a Network Emulator using Virtual Network Stack, *Proc. of the Seventh International Symposium on Operations Research and Its Applications (ISORA2008), Lecture Notes in Operations Research, Vol.8*, pp.351–358 (2008).