仮想ネットワークスタックを用いたGINEの機能拡張

M2007MM027 杉山裕介
指導教員：河野浩之

1 はじめに

広域ネットワークのアプリケーションの性能評価には，様々なネットワークエミュレータ/シミュレータが使用されている．GINE[4] は，多数のルータやリンクで構成される広域ネットワークをエミュレートすることができるネットワークエミュレータである．これは，IPv6 にも対応しており，dual core CPU での最大スループットは約 700Mbps に達している．しかし，現在の GINE には，パケット横取りに必要な Divert Socket を追加した非標準 Linux カーネルを用いる必要がある．また，ダイナミックルーティングやアプリケーションサーバ/クライアントのような実ルータ，実ホストを模倣できない．

そこで本研究では，パケット横取り方法を独自拡張カーネルの Divert Socket から Linux カーネル 2.6.14 以後標準の Netfilter NFQUEUE[3] に変更する．また，Linux カーネル 2.6.26 に組み込まれた仮想ネットワークスタックの実装である Network Namespace[2] を利用し，本物のルーティングデーモンを用いて自動経路設定を可能にする．機能拡張したネットワークエミュレータについて，中規模ネットワークモデルを例に，TCP, UDP 最大スループットの測定や RIP の起動などの基本性能を評価する．

2 GINEの概要

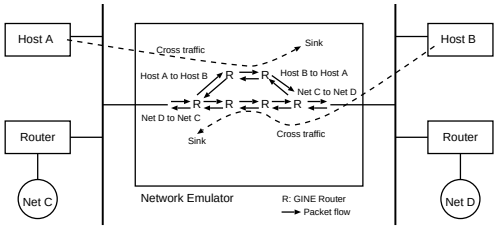

図 1 ネットワークトポロジエミュレーション

GINE では，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が可能である．GINE 内のリンクは，一般的なデータリンク層のフレームバッファとして使用される独自の Queue によって実現している．また，外部ホストまたは外部ネットからのパケットの送受信は，Divert Socket によるパケットの入出力で実現している．Divert Socket は，iptables で指定した Divert ポートに，指定されたプロトコルのパケットを横取り，再注入することができる特殊な socket である．

GINE では，図 1 のような，ルータとリンクからなるネットワークトポロジを模倣することができる．ここでのルータとは，GINE のプログラム内で作成された仮想ルータ (以下 GINE ルータ) である．また，GINE ではクロストラフィックを特定のリンクで与えることが可能である．

3 仮想ネットワークスタックの利用

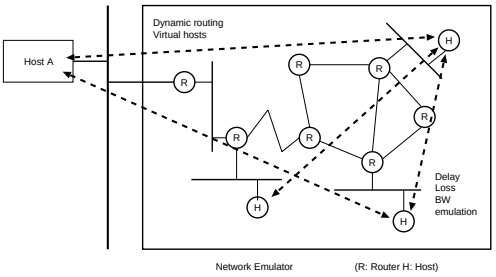

図 2 ダイナミックルーティングによるエミュレーション

図 2 のようなダイナミックルーティングモデルを GINE で構成するためには，ホスト/ルータの機能を GINE 内でエミュレートする必要がある．本来 OS で行われている処理を書き直すのは困難であるので，本研究では，仮想ネットワークスタックを用いる．その結果，実際のルーティングデーモンの使用やネットワークアプリケーションサーバ/クライアント (http など) を用いたエミュレーションが可能となる．

ネットワークスタックの仮想化とは，カーネルスペースで取り扱われるネットワークの機能を仮想化することである．OSI 参照モデルでいうデータリンク層の Device Driver(ネットワークディバイス) からネットワーク層全般の機能を仮想化する．これは Linux 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ため，比較的容易に使用することができる．

仮想化の結果，仮想ネットワークスタックとホスト間の通信では，OS のネットワーク機能が分離されているので，ループバックを含むネットワークインターフェイスを共有することができない．そこで，特殊な仮想ネットワークペアディバイスを用いる．このディバイスは，Linux 2.6.26 で Virtual ethernet pair device(以下 Veth) と呼ばれるペアーディバイスが存在し，片方のディバイスでパケットを受信するともう片方のディバイスに転送するしくみになっている．また，ループバックは自動生成される．

4 システムの実現

本節では，仮想ネットワークスタックの機能を含めた GINE の実装について説明する．

4.1 ソフトウェアコンポーネント

図 3 は，GINE のソフトウェアコンポーネントを表す．GINE のプログラムは，C++で記述し，マルチスレッドに C++プログラムと GNU common C++[1] ライブラリのクラスを用いる．

入力クラスである Input は，パケット到着待ちのスレッドとして単独して動く．Timer オブジェクトは，Condi-

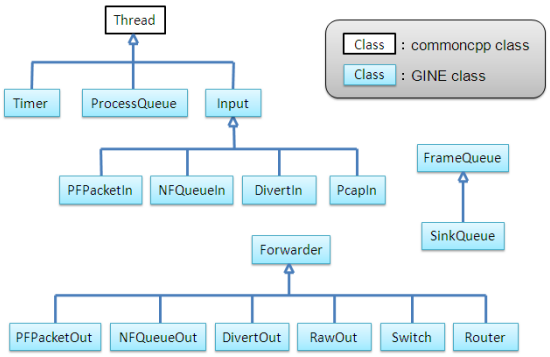

図 3 GINE のソフトウェアコンポーネント

tional というマルチスレッドの同期を測るインスタンスを共有し，ProcessQueue のスレッド同期に使用する．出力クラスである Forwarder はスレッドクラスではない．代わりに出力 FrameQueue を Timer で定期的に監視される ProcessQueue に登録する．そして登録された複数の FrameQueue のフレームが短い間隔で次の Forwarder に出力される．これによりスレッド数による CPU 使用率の増加を防ぐ工夫をしている．

4.2 NFQUEUE 入出力方法の追加

NFQUEUE 追加の理由は，Divert Socket は，独自拡張のカーネルが必要だが，NFQUEUE は，Linux 標準カーネルであるためカーネル実装が容易にできるからである．また，パケットフィルタリングは，カーネル標準の iptables を用いので，DivertSocket のように容易に Queue ターゲットを設定することができ，16bit(0 から 65535) まで区別できる．

図 4 に GINE に組み込んだ NFQUEUE 処理を示す．流れてきたパケットを GINE 内で作成した NFQueueIn で横取りして Queue に入れ，NFQueueOut で出力する．

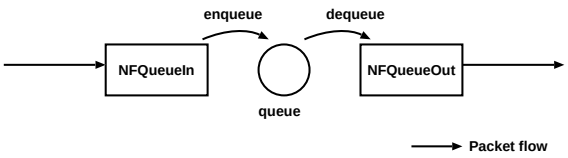

図 4 NFQUEUE の構成

NFQUEUE で横取りしたパケットは，NFQUEUE の独自ヘッダ+IP パケットで構成されている．よって，入力データとプログラム中で処理するデータが違うため，横取りしたパケットから NFQUEUE ヘッダを取り除いた IP パケットのみを取り出し，そのパケットを解析し，情報を記憶しておく必要がある．そして，解析したパケットのヘッダの DstIP アドレスで送信先へルーティングする．また，情報を記憶したパケットを入力に使った NFQUEUE に書き戻すため，その情報を記憶しておく必要がある．その際に定義した callback 関数を用いて NFQUEUE に書き戻す．

4.3 仮想ネットワークスタック間の通信

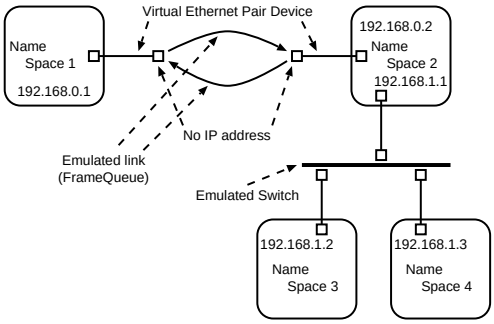

図 5 仮想 Namespace 同士の通信形態

図 5 で，仮想 Namespace 間の通信方法を示す．Name Space 1 と Name Space 2 の通信では，2 組の Veth の間に Queue を挟んで，仮想 Namespace 同士のリンク付けをしている．Queue への入出力には，データリンクフレーム入出力の PFPacket の Socket を用いる．また，エミュレータホスト側の Veth には，IP アドレスを割り振らない．これは IP アドレスがエミュレータホスト側に割り振られてしまうとエミュレータホストもしくはホスト自身の外側から届いたパケットは直接インターフェイスに転送され，横取りできなくなるからである．3 つ以上の仮想 Namespace の接続は GINE で作成したスイッチングハブを用いる．

4.4 仮想ネットワークスタックの自動生成

ネットワークスタックをホスト OS で手動作成する場合，1 つの仮想 Namespace に対して 1 つの端末が必要になる．これによりメモリの消費や，エンドユーザの操作を困難にする．よって，仮想 Namespace を GINE 内で管理し，1 つの仮想 Namespace 毎に様々な処理をさせる機能を追加した．これはスレッドの fork 関数で親プロセスと子プロセスを作り，pipe 関数を用いたプロセス間通信によりコマンド文字列の受渡しを実現している．

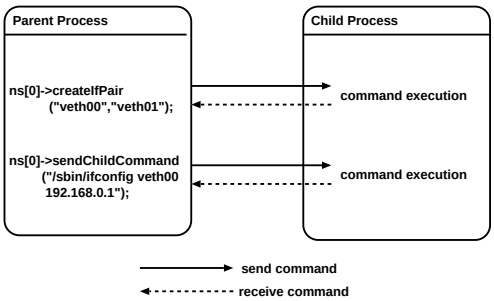

図 6 プロセス間通信によるコマンド送信

図 6 で示すように，親プロセスは，子プロセスの標準入力にコマンド文字列を送信し、実行結果を受け取る．子プロセスは，親プロセスから受け取ったコマンド文字列を実行し，実行結果を返す．Veth はホストのみで，生成や仮想ネットワークスタックへの受け渡しが可能であり，system 関数で実行できる．子プロセスでは，各端末で仮

想 Namespace の作成，RIP の起動などをコマンド文字列として受け取り，実行する．

4.5 GINE ルータと仮想ネットワークスタック間の通信

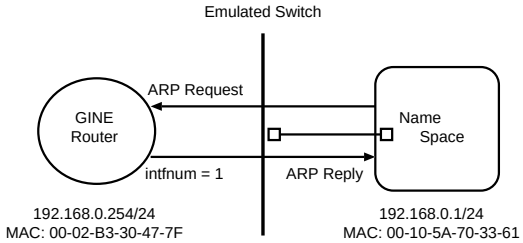

図 7 GINE ルータと仮想 Namespace 間の通信形態

GINE ルータには，IP アドレス，MAC アドレスを割り当てないため，実ホスト，仮想ネットワークスタックとの通信ができない．よって，図 7 で示すようにインターフェイス番号とは別に，IP アドレス，ネットマスク，MAC アドレスを設定可能にし，ARP(Address Resolution Protocol) 処理を実現した．

例として，仮想 Namespace から GINE ルータに ping をしたときの処理手順を示す．

1. 仮想 Namespace からスイッチにブロードキャストで ARP Request が送信される．
2. GINE ルータはユニキャストで ARP Reply を出す．
3. 仮想 Namespace から ICMP ECHO が GINE ルータに到着する宛先 IP アドレスで，ローカル処理関数に取り込む．
4. GINE ルータから仮想 Namespace に ICMP ECHO REPLY を送信する．この場合，初回は仮想 Namespace の情報が ARP テーブルにないので，仮想 Namespace の MAC アドレスを問い合わせる ARP REQUEST をブロードキャスト送信する．
5. 返事が届くまで待つ処理をする．
6. 仮想 Namespace から ARP REPLY が返ってきたら，宛先 IP アドレス (仮想 Namespace) の MAC アドレスの情報が ARP テーブルに記述され，仮想 Namespace との通信を可能にする．

また IPv6 での GINE ルータと仮想 Namespace 間の通信は，GINE ルータに IPv6 近隣探索プロトコル (IPv6 Neighbor Discovery Protocol) の機能を実装することにより実現している．

5 性能評価

性能評価に用いた PC は $\text{Intel}_{(R)}$ $\text{Xeon}_{(R)}$ CPU 1.86GHz，1000BASE-TX の NIC を 2 つ使用した．

5.1 パケット転送性能

ここでは，GINE に組み込んだ NFQUEUE 入出力方法を用いて，パケット転送性能を評価する．図 8 のようにネットワークエミュレータ内で GINE ルータを 1 台から 60 台までのそれぞれの台数で外部 Host A から外部 Host B に向けて，TCP, UDP 最大スループットを測定した．スループット測定には，iperf を用いる．

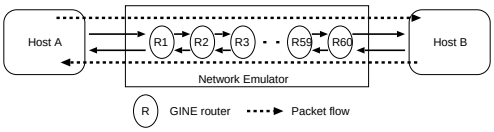

図 8 GINE ルータ直列エミュレーション

図 9 は，Queue の受信バッファサイズを 65535*4 バイトに変更した NFQUEUE と Divert Socket を測定した実験結果である．NFQUEUE の TCP，UDP スループットは，最大約 700Mbps と高速で通信することが確認できた．Divert Socket と比べて，UDP は高いスループットを示したが，TCP スループットは下回る結果となった．これは，NFQUEUE のプログラムの入出力処理の向上が必要である．

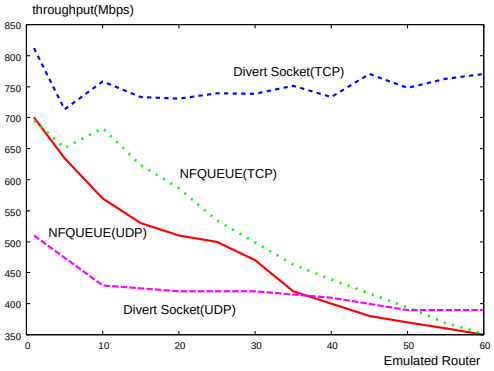

図 9 TCP, UDP スループット測定

5.2 GINE の動作確認

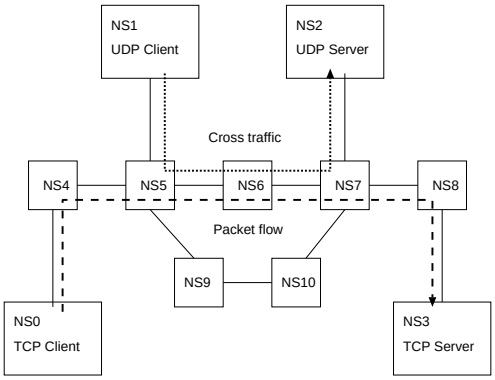

図 10 実験環境

図 10 のように仮想 Namespace 上で動作した仮想ルータ，仮想ホストを設置し，IPv4/v6 ネットワークを模倣する．IPv4 ネットワークは，それぞれ仮想ホストで RIP を起動し，IPv6 ネットワークは，静的に経路設定した．図 10 のように実験は，すべて NS0 から NS3 に ICMP パケットや iperf による TCP，UDP パケット送信で行い，以下のようなさまざまな実験を行った．

実験 1 ホップ毎の TCP, UDP スループット測定

実験 2 遅延，パケットロス，バンド幅制限の障害の発生

実験 3 クロストラフィックを流した場合の通信状況の確認

実験 4 RIP の起動確認

実験 5 仮想 Namespace 上で Web Server を起動

実験 1，2 の結果を表 1 に示す．実験 1 では，1 ホップで最大 161Mbps，UDP スループットは最大 146Mbps を測定している．スループットはホップ数に応じて減少するが，Queue へのパケット入出力やスイッチングハブのプログラムの修正により，スループットの低下を防ぐことができると考えられる．実験 2 では，仮想ルータで遅延，パケットロス，バンド幅制限などの障害を発生させ，その値と期待値を比較した．実験方法は，NS4-NS5, NS5-NS6, NS6-NS7 間でそれぞれ違うパラメータの障害を発生させ，NS0-NS3 間の通信により，複合障害の影響を測定する．遅延発生実験は，合計 600msec の遅延を発生させた．loss 発生実験は，各リンクで 10%ずつのパケットロスを発生させている．パケットロスが発生する期待値は，パケットロス値 x, リンク数 n の場合，$1-(1-x)^n$ で決まる．バンド幅は，NS0-NS4 間でバンド幅を設定し，iperf の TCP モードで実際のスループットを測定した．表 1 より各障害の影響を受けた結果は，期待値に近い値を示した．

表 1 GINE の動作確認

ホップ数 (n)	TCP(Mbps)	UDP(Mbps)
1 〜 3	71 〜 161	68 〜 146
4 〜 6	35 〜 63	28 〜 53
障害発生箇所	設定遅延 (msec)	結果 (msec)
NS4-NS5 間	100	-
NS5-NS6 間	200	-
NS6-NS7 間	300	-
total	600	606.599
障害発生箇所	設定パケットロス	結果
NS4-NS5 間	0.1	-
NS5-NS6 間	0.1	-
NS6-NS7 間	0.1	-
total	$1-0.9^3=0.271$	0.263
制限箇所	設定バンド幅 (Mbps)	結果 (Mbps)
NS0-NS4 間	1000	39.6
NS0-NS4 間	20	19.1
NS0-NS4 間	10	9.6

実験 3 では，図 10 のように NS1-NS2 間にクロストラフィックを流し，NS0-NS3 の通信状況を tcpdump を用いて確認した．バンド幅制限 30Mbps の NS0-NS3 間で TCP パケットを 100 秒間流している状態で，30〜40 秒，60〜70 秒の間，それぞれ 10Mbps，20Mbps の UDP クロストラフィックを発生させた．結果は，図 11 で示すように，NS0-NS3 の最大スループットは，クロストラフィックの量だけ概ね減少した．

実験 4 では，RIP を動作した状態で NS5-NS6 のリンクが途切れたら，経路が自動的に切り替わることを確認した．その結果，パケットが NS0-NS4-NS5-NS9-NS10-NS7-NS8-NS3 の経路で転送されるかを確認する．RIP は，routed を使用した．RIP は起動後，約 30 秒で経路情報を各ホストと交換し，全体に伝わる．RIP は，経路がなくなったさい，経路情報更新に約 120 秒かかる．実験を行った結果，約 120 秒で，通信が再開した．

実験 5 では NS3 上で Web Server を起動させ，NS0 から Web ページが閲覧できるかを確認した．Web サーバは，apache を起動することで実現しているが，各仮想ネットワークスタック毎に apache デーモンの設定ファイルを指定する必要がある．

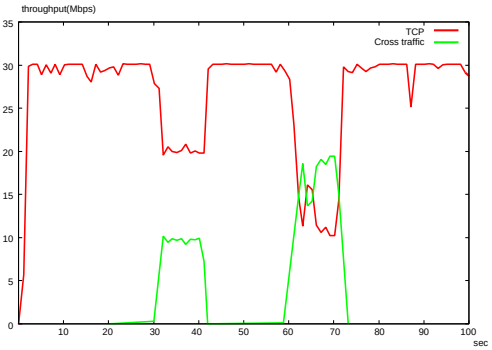

図 11 クロストラフィック測定結果

6 おわりに

本研究では，ネットワークエミュレータ GINE に仮想ネットワークスタックを組み込むことに成功した．これによって，さまざまなルーティングプロトコルを用いたダイナミックルーティングが可能になった．また，Netfilter NFQUEUE の追加により，標準カーネルでのパケット横取りを可能にし，カーネルの再構築が不要になった．パケット転送速度は，dual core CPU で約 700Mbps と十分速い．さらに，障害の模倣や RIP, apache デーモンの起動実験を行い，機能追加後の GINE が正常に動作することを確認した．今後の課題は，GINE の通信速度の向上と BGP を用いた広域ネットワークエミュレーションなどの応用例の追加，GUI の実装が必要である．

参考文献

[1] Free Software Foundation. Gnu common C++ http://www.gnu.org/software/commoncpp/, (accessed 2008).

[2] Network Namespace, “Linux Containers” (accessed Aug. 2008). http://lxc.sourceforge.net/network.php.

[3] Harald, W. : *libnetfilter_queue project* (accessed Nov. 2007). http://www.netfilter.org/projects/libnetfilter_queue/index.html.

[4] Ihara, A., Murase, S., and Goto, K. : IPv4/v6 Network Emulator using Divert Socket, *Proc. of 18th International Conference on Systems Engineering(ICSE2006)*, Coventry,UK, pp. 159–166 (Sep.2006).