無線障害を含むネットワークエミュレータの試作と評価

2003MT016 彦坂 将宏　　2003MT098 杉山 裕介
指導教員　後藤 邦夫

1　はじめに

近年，インターネットの普及に伴い，ほとんどの情報がインターネット上で扱われることが当り前になった．また，無線 LAN や携帯電話の普及に伴い，無線ネットワーク環境は急速に発展している．よって広域ネットワークのアプリケーションの性能評価では，無線ネットワーク環境も考慮する必要がある．有線ネットワークを模倣するエミュレータはいくつか存在するものの，無線環境下でのネットワーク障害を模倣するエミュレータは数少ない．

無線ネットワークを模倣するオープンソースのネットワークエミュレータには NCTUns[4] がある．ワイヤレス通信を含む様々なネットワークトポロジをエミュレートできるが，残念ながら IPv6 に対応していない．トポロジエディタなど GUI をもつ点で優れているが，最大スループットは 100Mbps 程度とそれほど高くない．

GINE（Goto's IP Network Emulator）[3] では，いくつかのルータを含む現実的なネットワークトポロジを C++ クラスライブラリを用いて簡単な C++ メインプログラムで実行することができる．50 台以上のルータを持つネットワークを 200Mbps 以上のスループットで模倣することも可能である．

本研究では，有線だけでなく無線を考慮したネットワーク環境を模倣し，障害パラメータとして無線障害を加えることでより現代のネットワーク環境に近づけることが一番の目的である．

GINE に無線障害を模倣する機能を追加することで無線ネットワークエミュレータを作成する．GINE を使う理由はプログラム追加変更が容易であり，IPv6 に対応している点で既存の障害模倣機構より優れているといえるからである．さらに GINE の改良と機能追加として，バグ修正，コードの整理，乱数発生の高速化，データリンク層での衝突回避モデル，QoS ルーティング処理，さらにユーザ機能として，GUI，統計情報レポートの機能追加などを進めていく．

なお，主に彦坂はカーネル修正を含む実験環境構築を担当し，杉山は無線モデル考案を担当した．プログラムの実装は共同で行った．

2　GINE のアーキテクチャ

図 1 はエミュレータ内部の構成とパケットの流れについて表している．外部ホスト A から送信されたパケットは iptables のルールにしたがって Divert Socket でパケットを横取りされる．Divert Socket とは iptables で指定した Divert ポートにパケットを横取り・再注入することが出来る．例えば，Host A から Host B へ送信されたパケットのプロトコルが ICMP である場合だけを Divert Socket で横取りするようにルール付けした場合，ICMP パケットは横取りされ，その他のパケットは素通りされる．

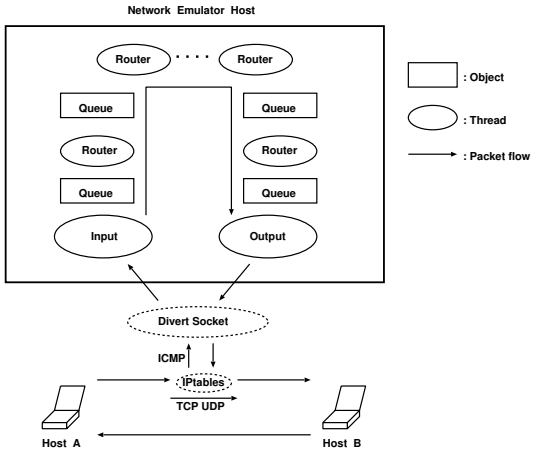

図 1　GINE のアーキテクチャ

図 2 は Host A から Host B への ICMP のパケット転送を Divert Port 8888 に横取りをする場合のコマンドラインである．（今回はホストからホストへの通信を例にあげているが，ネットワーク単位で指定することも可能である．）

```
# /usr/local/sbin/iptables -A FORWARD
  -p icmp -s HostA -d HostB
  -j DIVERT --div-port 8888
```

図 2　iptables のコマンド

横取りされたパケットは，最初の Queue に入り，指定した障害が与えられ次の Queue に渡される．それ以降も設定した n 個の数の Queue に次々に渡される．Queue での障害は，各 Queue 毎に違うパラメータの障害を起こすことが可能である．全ての Queue を通ったパケットは Divert Socket に再注入され，Host B に渡される．

今回は一方向のルータ直列のモデルを取り上げたが，ユーザによって，様々な目的に沿ってエミュレートすることも可能である．

3　カーネル修正

Divert socket は元々は IPv4 のみ FreeBSD カーネルに含まれている機能だが，Linux にも移植可能である．Linux カーネル用の IPv4 Divert Socket パッチは sourceforge.net で提供されており伊原らの研究 [2] で Linux 2.4 カーネルを IPv6 にも対応させた．

しかし，2.4 カーネルではデュアルコア CPU（もしくは 2 つの CPU）を搭載した PC で高速な通信をした場合、100 分の 1 より低い確率でパケットが divert されずに素通りしてしまうという問題があったので，本研究では SMP(Symmetric Multiple Processor) のサポートが向上している 2.6 カーネルを使用することにした．

2.6 カーネル用の Linux Divert Socket も IPv4 用のみの提供なのでカーネルソースを合計 1000 行程度の追加・修正をして IPv6 でも Divert Socket に対応させた．しかし，この 2.6 カーネルでも 2000 分の 1 程度と確率は低くなったものの，まだ divert 洩れの問題は残っている．また IPv4 用のパッチを当てただけでは IPtables のルールで OUTPUT のパケットを divert させるとカーネルパニックになってしまう問題が新たに出てきた．2.4 カーネルにあった OUTPUT の処理が 2.6 カーネルパッチには含まれていなかった．これを戻すことで解決した．

4　無線ネットワーク

無線通信の形態は，インフラストラクチャモードとアドホックモードに分けられる．アドホックモードはノード同士の通信なので狭義のネットワークであるのに対し，インフラストラクチャモードはアクセスポイントを介して，有線を含めた広域ネットワークに拡大できるので，本研究では，インフラストラクチャモードでの無線 LAN をエミュレートする方法を検討した．

4.1　無線障害モデル

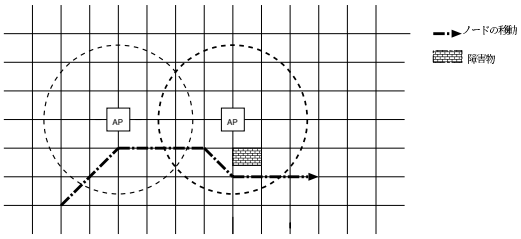

図 3　ノードとアクセスポイントの関係

本研究で考えた無線障害モデルは，複数のノードとアクセスポイントを 2 次元座標上に置き，無線障害を模倣するものである．初期設定として次の 5 点を定義する．

- ノードの数と初期座標
- アクセスポイントの数と初期座標
- 障害物の座標
- 座標の最大値
- 使用する無線 LAN 規格

座標の最大値を設定することによって建物の広さや部屋の広さを表現することができる．また，使用する無線 LAN 規格によってバンド幅とアクセスポイント-ノード間で通信可能な距離が決まる．

ノードの移動は 2 次元ランダムウォークによる移動モデルに従うか，手動で移動位置を決めることができる．図 3 は座標上にアクセスポイントと障害物，ノードの移動の関係を表したものである．アクセスポイントは同じ座標平面上に複数設置することができる．障害物もユーザが場所を指定することができる．ノードは図のように転々と移動する．移動してきたノードが円の中に入れば通信可能となり，また円の外に出たら通信不可ということになる．またアクセスポイントが複数ある場合は，ノードとアクセスポイントの距離の近い方のアクセスポイントを使う．(handover) また，アクセスポイントとノードの距離によってバンド幅を変化させる．距離が近付くにつれ電波が強くなり，バンド幅も大きくなる．距離が離れるとバンド幅は小さくなる．

4.2　無線障害モデルで用いる障害パラメータ

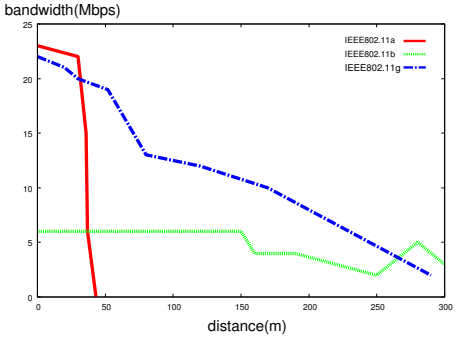

図 4　距離とバンド幅

無線では，電波強度により距離によって，バンド幅やパケットロス値が変化する．今回，実際の資料 [1] を参考にエミュレータに組み込むことによって，無線環境により近いものにしている．図 4 のグラフは無線 LAN 規格別の距離とバンド幅の関係である．無線 LAN 規格によって距離とバンド幅の関係は違うので，ユーザがどの無線 LAN 規格を使うかを選択する．距離に対してのバンド幅はより正確な値とするためにこのグラフを折れ線で表現し，その点と点との間を直線で補間することで求められる．

4.3　ハンドオーバー，障害物の模倣

ハンドオーバーとは，ノードが移動中に距離の近いアクセスポイントに切替えることである．切替えることにより通信が継続的に行われるが，アクセスポイントを切替える際に通信の途切れが生じてしまう．障害物は，2 点間を結ぶ線分を設定することで障害物の面積を表現することによりその影となる部分を障害対象範囲とする．

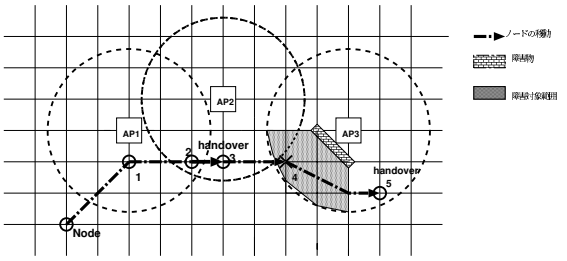

図 5　ハンドオーバーと障害物

この範囲内にノードが入った場合，遅延やパケットロスなどの障害を与える．図 4 では，ノードがハンドオーバーする場合とハンドオーバーする際の障害物の影響を表す．ノード (Node) が図のように移動し，1 の地点では AP1 を使用する．2 の地点では，AP1 と AP2 との距離が等しいので使用中の AP1 を継続して使用する．3 の時に AP2 の方が距離が短くなり，ノードは AP2 にハンドオーバーする．次に 4 で AP3 にハンドオーバーするのが本来の場合だが，AP3 を遮る障害物があるので，この場合は，AP3 へのハンドオーバーを止め，障害の少ない 5 の地点でハンドオーバーすることで通信障害を減らすよう設計した．

5　GINE の改良

この節ではカーネル修正以外の GINE の改良点について述べる．

5.1　統計情報レポート

ユーザ機能としてエミュレート終了時に統計情報を表示させるようにした．以下の情報を表示させることに成功した．

- エミュレートの経過時間
- 受信パケット数
- パケット損失率
- 各ルータ間でのパケット損失数，損失率の表示
- 無線障害でのパケットロスの値

エミュレータ側で表示することによって，ユーザにエミュレート結果がわかりやすくなる．

5.2　乱数発生の高速化

GINE では delay や loss の計算の際に乱数を発生させるが，その乱数発生に LinuxOS の random() 関数を使って乱数発生すると，再入可能でないためマルチスレッドプログラムでは不具合がでて、高速な通信の場合に計算が追いつかず速度低下も起こしてしまうことがわかった．乱数発生方法を各オブジェクトがそれぞれインスタンスを生成し，呼出しできるようにすることで再入可能にした．

確率分布に従う乱数発生には確率分布関数の折れ線近似で変換して出力している．乱数発生の高速化は確率分布関数の部分表の作り方を変えることで可能となった．

5.3　無線障害モデルの実装

本研究の無線障害モデルは，ノードが移動するたびに GINE で与える障害パラメータを単位時間毎に動的に変化させることによってノードと AP，障害物の位置関係による障害パラメータやハンドオーバーなどを模倣している．障害パラメータの変化は，GINE の一つの Queue だけで行っている．ノードの移動で，距離，障害物の障害対象範囲内か否か，ハンドオーバーする場合などのそれぞれの障害パラメータを Queue に与え，ノードが移動することにより Queue の障害パラメータの値を動的に変化させることによって，単位時間毎の無線障害を模倣している．

6　性能評価

実験に用いた PC は外部ホストとして Intel Celeron500MHz，256MB，1000Base-T NIC の PC を 2 台使用し，エミュレーションホストには Intel Pentium4(HT)2.60GHz，512MB の PC の HT（Hyper-Threading）ありとなしの 2 種類，Intel PentiumD 2.8GHz，1GB の PC，Intel Xeon 3.0GHz × 2，1GB の PC の 3 台 (全て 1000BASE-T の NIC)，計 4 種類の CPU で比較をした．

6.1　ルーティング性能

図 6 は一方向に仮想ルータを n 個直列につないだ場合のスループットを示す．

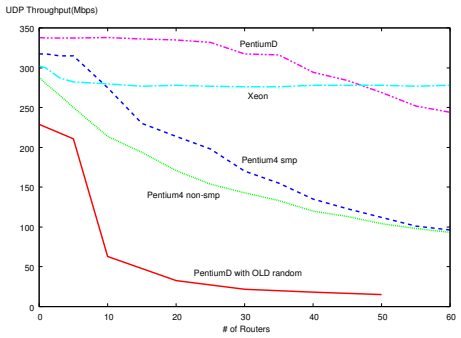

図 6　一方向 nRouters 直列エミュレーションのスループット

Pentium4 は予想通り HT を有効にした方が高い値が得られた．また，HT 無効にした場合はルータの数を 20 台に設定したあたりから端末の動作が重くなり，エミュレート中にキーボード入力を受け付けなくなってしまったが，HT ありではそのような事は起こらなかった．PentiumD はルータ 45 台までは一番良い結果となり，ルータ 50 台以上のネットワークでも 200Mbps 以上のスループットが得られた．Xeon では CPU の特性通り高負荷時にも安定した動作をし，ルータの数を 50 台以上に増やしても高いスループットが得られた．

また，図 6 の実線は乱数発生方法変更前の PentiumD の結果である．この結果から，乱数発生方法を変更したことによりスループットが向上していることが分かる．

6.2 無線エミュレーション

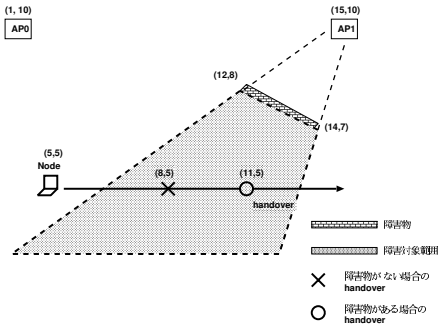

図 7　無線エミュレーション記述例

図 7 で示すように，2 つのアクセスポイント (AP0,AP1) とノード (Node) と障害物を用いて実験した．ノードの初期座標は (5,5) の位置を取り，手動で x 軸を並行に (6,5) → (7,5)…と進めていくよう設定した．移動時間は 1 マス移動するのに 10s と設定した．アクセスポイントや障害物の座標は図 7 に示す通りである．

```
 1 AP0=(1,10) AP1=(15,10) NODE0=(5,5)
   IEEE802.11b
 2 NextAP=AP0 Measure:6.40 PacketLoss=0.00%
 3 NextAP=AP0 Measure:7.07 PacketLoss=0.00%
 4 NextAP=AP0 Measure:7.81 PacketLoss=0.00%
 5 NextAP=AP0 Measure:8.60 PacketLoss=0.00%
 6 NextAP=AP0 Measure:9.43 PacketLoss=0.00%
 7 NextAP=AP0 Measure:10.2 PacketLoss=0.00%
 8 NextAP=AP1 Measure:9.60 PacketLoss=0.39%
   handover
 9 NextAP=AP1 Measure:8.74 PacketLoss=0.00%
10 NextAP=AP1 Measure:8.07 PacketLoss=0.00%
11 NextAP=AP1 Measure:5.09 PacketLoss=0.00%
//中略
16 NextAP=AP1 Measure:25.4 PacketLoss=0.00%
17 NextAP=AP1 Measure:26.4 PacketLoss=0.40%
18 NextAP=AP1 Measure:27.4 PacketLoss=1.00%
19 NextAP=AP1 Measure:28.4 PacketLoss=0.80%
20 NextAP=AP1 Measure:29.4 PacketLoss=0.99%
21 NextAP=AP1 Measure:30.4 PacketLoss=2.80%
22 NextAP=AP1 Measure:31.4 PacketLoss=7.20%
```

図 8　無線エミュレーション実験結果

図 8 は上記の設定で行った実験の結果である．NextAP=AP0 は移動後に使うアクセスポイント，Measure はノードとアクセスポイントの距離，PacketLoss は距離によるパケットロス値を表している．1 行目は初期設定値で，無線規格は IEEE802.11b を使用する．2 行目からがノードの移動を表している．本来 4 行目の地点でノードはハンドオーバーするが，障害物の影響で，7 行目でハンドオーバーが起こっている．これは障害物の影響を受けるアクセスポイントは，ノードとの距離を本来の距離より伸ばすことで実現している．ハンドオーバーは，50ms 間パケットロス 100% なので，移動時間 10s と合わせて 10.05s 間中の 50ms パケットロス 100% ということになり，ログでは，PacketLoss=0.39% と表示される．10 行目に障害対象範囲から抜け出すので，距離が短くなっている．そして 26m 辺りからパケットロスが起こり始めている．この実験から設計通り無線エミュレータが動作することを確認した．

7　おわりに

本研究は，ネットワークエミュレータである GINE の改良および機能追加が目的である．無線障害を発生するためにインフラストラクチャモードでの障害モデルを考案し実装した．2 次元座標上にノードとアクセスポイント，障害物を設置し，距離によるスループットの低下やハンドオーバーによる通信の途切れなどをエミュレートすることが可能になった．他にも，SMP カーネルでの divert 洩れを防ぐために 2.6 カーネルを IPv6 Divert Socket に対応させ，エミュレータホスト側での統計情報機能を追加し，乱数発生の高速化をし，複数ルータエミュレーションなどでの最大スループットが向上した．

今後の課題として，スレッドコントロール方法の変更などによる更なる高速化，IPv6 固有の QoS ルーティング処理，データリンク層での衝突回避モデル，GUI の実装などが挙げられる．同じアクセスポイントを使う他の端末の存在による障害など，詳しい障害パラメータを組み込み，より現実的な無線ネットワークを模倣することが望まれる．また，インフラストラクチャモードとアドホックモードを組み合わせたメッシュネットワークのエミュレーションに発展していくことを期待したい．

参考文献

[1] Basagni, S., Conti, M., Giordano, S. and Stojmenovic, I.: *IEEE 802.11 Ad Hoc Networks: Protocols, Performance, And Open Issues*, chapter 3, pp. 69–116, IEEE (2004).

[2] 伊原亜希，村瀬進哉：ネットワーク障害の模倣を目的とする仮想ルータの設計と実現，卒業論文，南山大学 数理情報学部 情報通信学科 (2004).

[3] 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).

[4] Wang, S. and Chou, C.: Innovative Network Emulations Using The NCTUns Tool, *Computer Networking and Networks* (Shannon, S., ed.), Nova Science Publishers, chapter 7, pp. 159–189 (2006). (`http://nsl10.csie.nctu.edu.tw/`).