性能評価のための仮想ネットワークの構築

2000MT032 伊藤 洋介　2000MT059 中本 拓也　2000MT078 大藤 純一　2000MT111 吉田 秀考
指導教員　後藤 邦夫

1　はじめに

インターネット基盤は、今後も継続する接続機器と通信量の増加へ柔軟に対応できなくてはならない。さらに、利用の多様化を背景としたデジタル通信の質およびサービス機能を実現し、信頼性のある産業・生活基盤としての役割の確立が急務である。

しかし、既に構築済のネットワークへ実験または試験段階の装置やソフトウェアを接続し、性能評価を行うことは障害を起こし得る可能性がある。過去に向井らが行っていた研究 [1] では、IPv4 における仮想ネットワークを構築し、ユーザプロセスでルータを模倣して故意に通信障害を起こして性能評価を行っていた。この研究では、障害の要素が現実のネットワークで想定されるものと比べて少なく、十分にテスト対象の性能を評価できない。

そこで本研究では、より実際のものに近い仮想ネットワークを構築し、性能評価を行う。実現の方法は、向井らの研究と同様、Divert Socket と IP firewall を用いてルータを模倣し、ルータを通過するパケットに障害を与えることによって広域ネットワークを模倣することである。そして、先の研究では実装されていなかった障害の要素を加え、実際に発生し得る問題を想定できるようにした。この研究によって、運用中のネットワークに接続することなく、ネットワークアプリケーションの実験・テストを行う環境の提供が可能になる。

本論文では、まず仮想ネットワークの概要について、構成する機能と、ネットワークに考えられる状況や問題点を示し、次節では仮想ネットワークの実際の構築方法について必要な条件を整えるためのプログラムの説明をする。次にこれらを用いて、擬似的に通信障害を発生させ、指定通りの通信障害が起こせるか評価を行う。なお、吉田はネットワークの基礎構築部分、伊藤は Divert Socket、 IP firewall、IPv6 について、中本は各障害発生プログラム、大藤はキューイングと障害発生の確率分布を主に担当した。

2　仮想ネットワークの概要

仮想ネットワークを構築する手段は次の通りである。まずユーザプロセスで仮想ルータを実現させ、そのルータを通るパケットを Divert Socket を用いて横取りし、目的アドレス別にパケットを振り分け、それぞれに対しルーティングを行う。用意したルートによって与える障害を変え、行き先別に違いが出るようにする。

2.1　ネットワークモデル

今回の実験に使うハードウェアは、ルータとなる PC1 台とホストとなる PC 数台である。ルータ用の PC の OS は Divert Socket を利用するために FreeBSD にする。モデルの具体的な内容は以下の通りである。

1. ホストは、セグメントの異なるネットワークにそれぞれ配置
2. セグメントの異なるホスト同士で通信を行う (ルータ機能を使わせるため)
3. ルータを通過するパケット全てをファイヤーウォールによって Divert Socket に渡す
4. Divert Socket からパケットを横取りし、プログラム内でパケット毎に処理を実行する
5. 再注入する。その後は 4-5 のループ

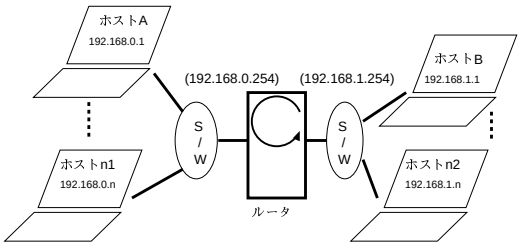

図 1: ネットワークモデル

2.2　並行処理

仮想ネットワークを実現するにあたってプログラムが伝送経路に相当するが、これが通常のシングルスレッドの場合、処理の流れは 1 本であるためにパケットを受け取る部分と、パケットを処理し Divert Socket に再注入する部分の独立ができない。これでは、障害を与える上で影響を受けてはならない箇所も影響を受けてしまう。そこで作成するプログラムは、並行処理できるようにすることが必要になる。並行処理の方法として、コピープロセスを生成させるものと、プロセスは 1 つで中にスレッド (軽量プロセス) を生成するものがある。プロセスを増やす前者の方法では、システムにかける負担が多くなり、メモリを多量に消費する恐れがある。これは、個々のプログラムを同時に起動させて実験を行う上で支障が出る可能性がある。後者の場合、プロセスは増やさ

ずに処理をこなすことが可能なので、そういった心配は軽減される。

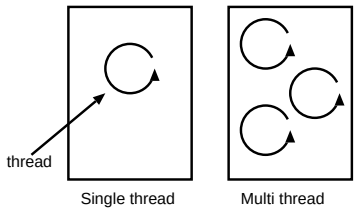

図 2: シングル・マルチスレッドの違い

本研究ではマルチスレッドを用いたプログラミングを行うことにした。

2.3 ネットワーク上で発生する問題

現在、インターネットは多数のネットワーク同士が接続することによって構成されている。しかし、接続すべきネットワーク数が増えるにつれ、接続形態が非常に複雑になり様々な問題点が発生しており、それを解決する仕組みが考えられている。ネットワーク形態は特定のポイントを経由する場合が多く、そこに依存する形が多いため以下のような問題点が発生する [2]。

通信遅延の発生

通信品質を悪くする大きな要因は遅延の問題で、その要因としては、ハードウェアの処理に要する時間およびネットワーク伝送遅延がある。

通信の損失

パケット損失とは、予期できない原因によりパケットが途中で行方不明になり送り先に届かないことである。

通信経路、ハードウェア障害

その他の障害として、ネットワーク・カードやケーブル，ハブなど物理的な障害、どこかの接続で不良個所があるか、ケーブルが断線しているなど、ハードウェアの故障などが考えられる。

3 仮想ネットワークの実装

この節では、実験環境で帯域制御、伝送遅延、損失の実現方法を説明する。

3.1 複数ルータの実現

実際のネットワークというのは数多くのルータを介して通信が成り立っており、それぞれのルータの性能や置かれている環境によって通信性能が大きく変わってくる。そこで本研究では、一つの仮想ルータにおいて複数のルータを実現させるために IP firewall のルールを用い、一つのルールで一つの障害を起こし、それぞれをマッチングさせていくことで複数の障害を模倣することにした。それにより複数ルータを実現することができた。

IP firewall のルールにかかり divert socket で divert されたパケットは、処理を終えた後再注入されるが、既にかかったルールは飛ばして次のルールにいくことができるためループしてしまうことはない。しかし IP firewall は、入口と出口の二つのインターフェースにおいてそれぞれ適応するため、ルールを追加するときにインターフェースを指定しないと２回ルールにかかってしまう。

3.2 Pthread による並行処理

マルチスレッドプログラミングとして、POSIX 1003.1c-1995 標準に準拠したスレッド・ライブラリを使う Pthread を利用する。Pthread には、スレッドの生成から消滅までに必要な API が用意されており、プログラム作成において扱いやすいものになっている。

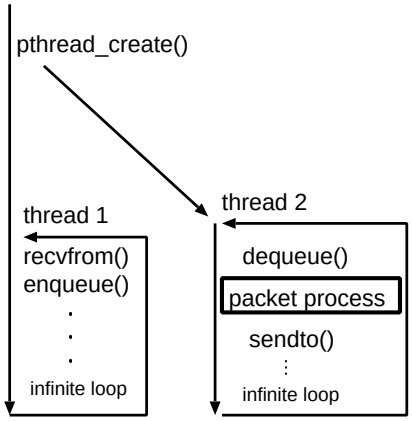

図 3: 並行処理プログラムの流れ

3.3 無限ループ防止について

再注入されたパケットは、再び IP firewall の位置に戻される。しかしこのままでは、パケットは再び Divert Socket に渡されるルールを通過してしまい、パケットが無限ループに陥ってしまう。それを防ぐために Divert Socket は、プログラムでパケットを受け取った際に取得できるアドレス情報に特殊なデータを添付する。この添付されたデータによって IP firewall のルール (パケットが Divert Socket に渡された時のルール含む以前の) をスキップできる。この仕組みによって、無限ループの防止が実現されている。

ソケット入出力の際、recvfrom() で得たアドレス情報は全て保存する。これは、パケット再注入を正常に行うためである。無限ループ防止の為にも必要である。また、アドレス情報の長さを格納する変数を初期化 (sizeof() を用いる等) を行わずにいると、再注入不能に陥いる [3]、[4]。この手順を怠ったことで、パケットを受け取ったものの、再注入できないトラブルに暫く見舞

われた。

3.4 タイマについて

プログラムで使用するタイマとして当初 usleep() を使用したが、正確に処理の流れを止めることができなかった。そこで、gettimeofday() をループさせるルーチンを組み、目的に近い時間だけ止められるようになった。

また、myusleep() を使うことで短い時間 (マイクロ秒) での処理を実現させるようにした。

3.5 伝送遅延、損失等の実装方法

この節では、多種多様な IP パケットの伝送経路を用意し、その経路上でパケットの損失、遅延の違いを観察できるようにシミュレートする方法を考える。伝送経路を用意するというのは、パケットに応じて処理方法を変えることであり、各伝送経路を振り分けるのに本研究では宛先 IP アドレスを用い処理方法を変えるようにした。

4 性能評価

仮想ネットワークを構築した上で、製作したプログラムを使い障害を発生させ、収集したデータと確率関数と比較し評価を行う。

4.1 確率分布に従う障害単体での性能評価

各障害を単体で発生させ、それらがモデル化した各確率分布に従って発生しているか、評価を行う。

一定遅延

一定遅延を測定するのに ping を使った。送信パケット数を 10000 に設定し、0.01 秒間に 1 個の間隔で ping を送るようにした。

表 1: 一定遅延 (ミリ秒)

遅延時間	最小遅延	最大遅延	平均遅延
100	99.088	143.514	113.907
1000	997.456	2060.003	1012.839

ルータ内でパケットを処理しているため多少の誤差はあるが、目標とする遅延を発生させることができた。

ランダムロス

・一様分布に基づくランダムロス

ランダムロスを測定するのに ping を使った。その設定として送信パケット数を 10000 に設定し、0.01 秒間に 1 個の間隔で ping を送りその損失具合を観測した。

表 2 にその結果を示す

パケット数が多いほど、より指定した確率に近い損失が発生することが分かった。

次に計測したパケットの中から、連続した 200 個のパケットを無作為に取り出し、損失率を比較した。

若干の誤差はあるが、パケット損失具合は一様に分布

表 2: パケット損失率

パケット数	パケット損失数	損失率 (%)
10000	513	5.13
100000	5022	5.02

表 3: パケット損失の分布

シーケンス番号	パケット損失数	損失率 (%)
101-300	10	5
3101-3300	11	5.5
5501-5700	8	4
9001-9200	10	5

しているという結果を得ることができた。

パケット順序入れ換え

パケット順序入れ換えを測定するのに ping を使った。そして ECHO_RESPONSE のシーケンス番号を見て、パケット順序が入れ換わっているかを観測した。入れ換わる確率やそのときの個数の平均などを様々に変えた。そのデータを表 4 に示す。

表 4: パケット入れ換え

入れ換え確率 5% 平均数 3	
入れ換わった長さ	発生回数
20	1
18	1
17	1
…	…
3	58
2	82
1	131
総入れ換え数	426
発生確率 (%)	4.8

スループット制御

スループットを測定するのに FTP を使った。ファイルサイズの大きいものや小さいものをホスト A からホスト B へ転送したときのスループットを観測した。

表 5 にその結果を示す。

パケット重複

一定遅延を測定するのに ping を使った。その設定として送信パケット数を 10000 に設定し、0.01 秒間に 1 個の間隔で ping を送る。パケット重複確率 (一様分布) と重複平均数 (指数分布) を様々に変え、そのデータを観測した。

表 5: 206MB のファイル

制限	1MB/s	500KB/s
実測値	940KB/s	408KB/s

表 6 にその結果を示す。

表 6: 重複確率 20% 重複平均数 3

重複した数	同じ数重複した個数
20	2
19	2
18	3
…	…
3	180
2	302
1	390
総重複数	1473

4.2 複数の障害での性能評価

各障害を複数発生させ、より実際のネットワークに近い状況を作り、様々な状況下での性能評価を行う。
三つの障害を組み合わせた状態での性能評価
・バーストロス+パケット順序入れ換え+パケット重複
ipfw のルールを追加

```
ipfw add 100 divert 12345 ip from
 192.168.0.0/24 to any via xl0
ipfw add 200 divert 12355 ip from
 192.168.0.0/24 to any via xl0
ipfw add 300 divert 12365 ip from
 192.168.0.0/24 to any via xl0
ipfw add 400 allow ip from any to any
```

バーストロスとパケット順序入れ換えとパケット重複のプログラムを実行

```
$ ./burstloss 100 20 3 12345
$ ./constdelay 100 20 3 12355
$ ./duplicate 100 10 5 12365
```

各性能評価を行ったが、遅れのプログラムで指示した値と若干差が出る結果となった。予想される問題点としては処理の負荷が大きくなり、一方のプログラム、スレッドに偏ってしまったことである。ロス関連のプログラムについては、ほぼ指示通りの結果が得られており、複数の障害が重なった通信路を再現できた。

5 おわりに

本研究では、通信性能を評価するために、仮想ネットワークを構築し、実験を行った。実際のネットワークを模倣するにあたって、経路途中でのパケット損失、伝送遅延を実現するために、Divert Socket, IP firewall を用いて仮想ルータを作成し、ルータを通過するパケットに障害を与え、評価を行った。障害は様々な確率関数に基づいて我々が独自に作成したプログラムによって実現した。実験結果から、収集し解析したデータは確率関数モデルに従っているという結果が得られた。また、IP firewall,Divert Socket のルールを応用し、複数ルータを模倣することも成功した。一つのルールで一つの障害を起こし、組み合わせることで様々な複数の障害を発生させることができた。

しかし、Divert Socket の設定方法や、IP firewall の無限ループ問題、ip6fw の Divert Socket 未対応問題などに時間を取られ、研究がなかなか進まなかった事が残念でならない。

今後の課題は以下の通りである。

1. 個々の独立したプログラムを一つのプログラムにする
2. IPv6 のフローラベルを用いたフロー制御の実現

特に前者の課題を解決して初めて仮想ネットワークを完成したと言える。

参考文献

[1] 向井 大樹, 仙田 裕二, “性能評価のための仮想ネットワークの構築”, 南山大学経営学部情報管理学科卒業論文, 2003.

[2] Paul Ferguson and Geoff Huston, “インターネット QoS”, オーム社, 2000.

[3] W. Richard Stevens, **UNIX ネットワークプログラミング Vol1 ネットワーク API、ソケットと XTI**, ピアソン・エデュケーション, 2000.

[4] W. Richard Stevens, **UNIX ネットワークプログラミング Vol2 IPC プロセス間通信**, ピアソン・エデュケーション, 2000.