既存のフィードバックを用いたIP複数経路伝送システムの改良と性能評価

2005MT052 小林 愛司　　2005MT118 寺尾 充弘
指導教員　後藤 邦夫

1　はじめに

近年，インターネット利用者が増加し，動画像の配信サービスなどのオンラインコンテンツが増えてきたため，ネットワーク全体におけるトラフィックが増加している．しかし現在のインターネット環境は，単一経路での通信をしているため，利用者が多い場合に利用可能な帯域幅を占有してしまい，ネットワーク全体のスループットの低下や，パケットの損失や遅延などの影響が出てしまう．その解決策として同一のデータを複数の経路に振り分けて伝送を行うことにより，各経路ごとにおける負荷を減らし，利用可能な帯域幅を確保することが可能である．

これまでに，[2][3][5][6] のような複数経路の伝送についての先行研究があり，主にアルゴリズムの提案，伝送方式などがあり，その結果をネットワークシミュレータやネットワークエミュレータでの性能評価を行っている．

そこで本研究では，先行の研究の中でパケットの配送効率の高く，経路の輻輳状況をフィードバックし，経路の混雑状況に応じて，経路ごとにパケットの割り振りを変えて伝送効率を上げる [2] のアーキテクチャの通信速度を更に上げるためのシステムの改良を目的とする．そしてネットワークエミュレータを使用し，最大 10 経路での性能評価実験を行い，経路数の多いときの複数経路伝送の有効性を示す．

小林は主にプログラムの改良，寺尾は主にネットワークの構築と性能評価実験を行った．

2　既存の複数経路伝送システムの概要

本研究では，ネットワーク上で複数経路にパケットを振り分けて配送する．前提条件として，アクセス回線は十分速いものとする．例えば，複数 ISP のサービスを契約して複数経路を利用できる場合などを考える．図 1 のように送信ノード (HostA) と受信ノード (HostB) の間に，2 つの中継地点である中継ノード (以下，Relay) を設置する．そしてネットワーク上に設置された RelayC, RelayD に向けて両端ホストがデータを配送することで，複数経路でのパケットの配送が可能になり，図 1 の矢印のようにパケットが流れる．この仕組みにより片側の経路に多くのトラフィックが流れていて，使用できる帯域幅を占有してしまっている状態でも，他の経路が用意されているため，データの配送の効率を下げずに済む．

両端ホストでは，Relay にパケットを振り分ける既存のプログラムを起動する．IP パケットを複数の経路に振り分けるために，各経路の遅延情報をフィードバックし，その情報をもとに計算を行い，送信する経路を決定する．経路選択に使用する振り分けアルゴリズムは第 3 節で説明する．経路設定では，UDP トンネリングを用い，トンネルデバイス (以下，tun(デバイス)) を tunctl コマンドを用い，ネットワーク上の両端ホストの 2 点間で IP アドレスを指定し，パケットをカプセル化し通信を行う．

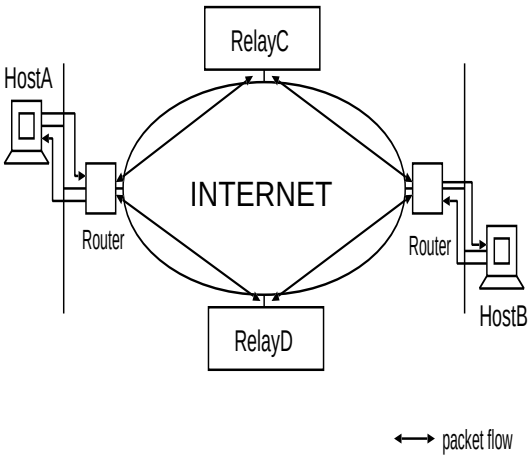

図 1　ネットワークモデル

2.1　プログラムの構成

提案された複数経路伝送のプログラムの構成要素について説明する．このシステムは図 2 のように，5 つのクラスから構成され，それぞれ両端ホストで起動する．

まず Tunnel interface に入ってきたパケットを Xmit で読み込む．この Xmit では，RecvEnqueue から受信した経路の混雑状況に応じて，パケットを効率良く配送するために，経路ごとに重みをもとにして，データを配送する最適経路を選択する．そして，UDP socket を利用し，カプセル化されたパケットを選択された経路によって送信先ホストへパケットを配送する．

RecvEnqueue では，両端ホストがともに各経路の混雑状況を知るために，時間情報を周期的にフィードバックする．その情報から各経路ごとの遅延の値を利用し，各経路の混雑状況を提案された振り分けアルゴリズムの計算式で算出し，その算出された経路の重みを Xmit に渡す．フィードバックされた時間情報以外の送られてきたデータは， PacketQueue に送られる．

PacketQueue では，トンネリングヘッダに含まれた連

続番号の順序で，queue にパケットを格納する．パケットごとに連続番号が割り振られているため，複数の経路にパケットを送信した際に，各経路ごとの送信速度の違いにより，その到着順序が変わってしまっても，その並び変えが可能であるため，送信したパケットの順番通りにデータを配送できる．

DequeueSend では，queue にパケットがあるかどうかを周期的に検査する．もし queue にパケットが格納されていたら，ペイロードをカプセル開放し，Tunnel interface にデータを書き込む．

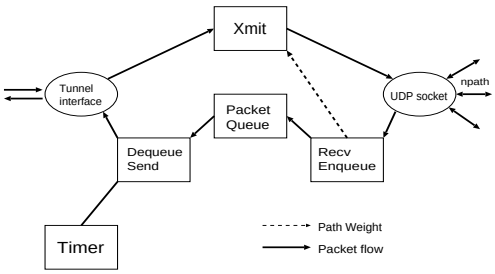

図 2　プログラムの構成要素

2.2　システムの時刻同期

このシステムでは，時間情報から各経路の遅延を計算しているため，システムを使用する両端ホストで時刻同期を行う必要がある．両端ホストの時間が相対的にあってない場合，各経路の遅延情報を正確に得ることができないため，経路選択に使用する計算に影響が出てしまう．複数経路伝送システムを使用する両端ホストを HostA, HostB とし，以下のどちらかの方法で時刻同期させる．

- 参照できる NTP サーバがないとき，HostA の NTP デーモンは自分の時計にあわせ，HostB の NTP デーモンは HostA の NTP デーモンにあわせる．
- 参照できる NTP サーバがあるとき，HostA と HostB それぞれ外部 NTP サーバにあわせる．

3　振り分けアルゴリズム

本節では，[2] で提案された振り分けアルゴリズムについて説明する．これは，両端ホストで相対的に時刻があっていることが前提である．この振り分けアルゴリズムの計算は各経路ごとにそれぞれの最小遅延, 最大遅延, 相加平均遅延の値をもとにして経路ごとの重みを計算するものであり，計算式は以下のようになっている．

$$D_i(t+1) = \mathrm{MeanDelay}_i(t+1) - \mathrm{MinDelay}_i(t) \quad (1)$$

D：混雑推定値
t：時間
$\mathrm{MeanDelay}_i$：相加平均遅延
$\mathrm{MinDelay}_i$：最小遅延
d_i：混雑推定値の逆数の比
W_i：d_i を用いた path i の重み
α：0 から 1 の敏感時間パラメータ

$$d_i = \frac{\frac{1}{D_i(t+1)}}{\Sigma_i \frac{1}{D_i(t+1)}} \quad (2)$$

$$W_i(t+1) = (1-\alpha)W_i(t) + \alpha d_i \quad (3)$$

- (1) 式は，経路が混雑している状態を値として示す．各々の経路の相加平均遅延と最小遅延の差をとる．この値をとることで，遅延の増加分のみを重みとすることができるので，遅延の違いによる振り分け配分の不公平を減らすことができる．
- (2) 式は，混雑推定値 d_i が小さい経路を優先するため．
- (3) 式は, 時間 t で経路 i のときの重さ $(W_i(t+1))$ を示す．α で重みをゆるやかに更新することで，重みの急激な変化を防ぎ，ジッタを小さくすることができる．

4　既存の複数経路プログラムの高速化

本節では，本研究で用いる既存の複数経路プログラムに高速化ついて述べる．

4.1　Xmit の改良案

変更すべき箇所としては，データ転送の役割を持つ Xmit である．既存のプログラムでは Xmit 内で送受信の処理と経路設定を行っている．プログラムの構造上，この構造では送信と受信の処理を同時に行うことができないため，配送効率が悪くなっている．

そこで図 3 のように Xmit ではパケットの受信と経路設定だけの処理にする．そして読み込んだパケットを PaketQueue に格納し，DequeueXmit で送信の処理をする．このようにすることで，送信と受信を並列処理させることが可能となり，効率良くデータ配送することができる．

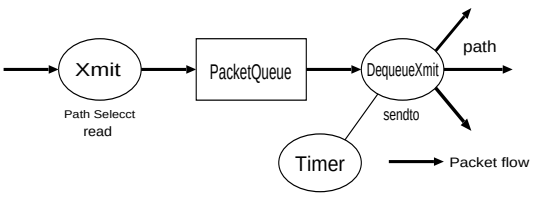

図 3　Xmit の改良案

4.2　Relay の改良前

現在の Relay プログラムでは，socket 1 つで通信を行っている．図 4 のように両端ホストを HostA, HostB としたとき，HostA から HostB，HostB から HostA

の通信を 1 つの socket で処理する構造となっている．プログラムで指定するものは，HostA の IP アドレス，HostB の IP アドレス，Relay の IP アドレス，Relay の使用するポート番号である．

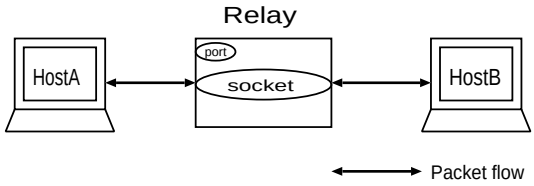

図 4　Relay の改良前

4.3　Relay の改良案

Relay を高速化する手段は，通信で利用する socket を 2 つにし，それぞれ別のポートで受けとるようにする．このようにすることで，図 5 のように, HostA から HostB への通信を socket 1, HostB から HostA への通信を socket 2 というように，各 socket の入出力をマルチスレッドで並列処理させることで Relay を高速化できると考えた．プログラムを使用する際は，socket を 2 つ使用するため，Relay で使用するポート番号も 2 つ使用しなければならない．

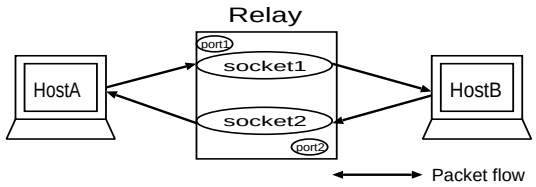

図 5　Relay の改良案

5　実験

本節では，本研究における実験環境と性能評価実験の詳細について述べる．

5.1　実ネットワーク実験の断念

当初，私達はネットワークエミュレータではなく実ネットワークでの複数経路伝送での実験を考えていた．しかし，実験に協力してくれる多くの人が ADSL を使用していた．ADSL は下りの速度は十分でも上りが遅いため，実験に必要な速度を確保できないことがわかり，実ネットワークでの実験を断念した．実ネットワークでの実験では，実際に実ネットワーク上で複数経路伝送システムを使用し，通信確認をすることができた．

5.2　ネットワークエミュレータ実験環境

本研究では既存のネットワークエミュレータである Goto's IP Network Emulator(以下，GINE)[1] を使用する．図 6 に 2 経路の実験環境を示す．

GINE を用いる場合の実行手順を以下に示す．

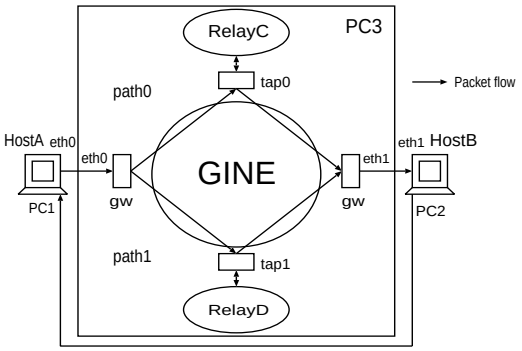

図 6　ネットワークエミュレータ実験環境

1. HostA, HostB, GINE を用いる PC の eth0, eth1 にプライベート IP アドレスを設定し，デフォルトゲートウェイで接続する．
2. GINE を用いる PC で，タップインターフェースを送信したい経路の数だけ作成し，同じ数の Relay プログラムを起動する．
3. HostA，HostB で複数経路伝送プログラムの起動と時刻同期を行う．GINE の構造上 HostA から HostB は任意の経路数，HostB から HostA は単一経路に設定する．
4. 各経路の帯域幅制限，遅延の設定した GINE を起動する．

5.3　スループット計測

ネットワーク帯域幅測定ツールである iperf を使用し，GINE を利用して TCP スループット, UDP スループットを測定する. 実験は単一経路と複数経路で行う．

■複数経路伝送システムの最大スループット　システムの TCP と UDP の最大スループットを計測するため，帯域幅制限のなしの単一経路で実験を行う．表 1 は実験結果である．

表 1　最大スループット

	スループット [Mbps]
TCP	85.1
UDP	87.5

■両端ホストが時刻同期を行わなかったときのスループット　両端ホストの時刻同期を行わず，最大 10 経路の複数経路で，各経路の遅延を 0.0[sec]，帯域幅制限 10[Mbps] に設定し，TCP スループットと UDP スループットを測定した．表 2 は実験結果である．

実験結果より，システムの時刻同期を行わなかったとき，TCP スループット，UDP スループット共に，経路数を 3 経路以上にしたときに，期待通りの結果を得ることができなかった．この原因は，両端ホストに時間の

表 2　時刻同期を行わなかったときのスループット

経路数	帯域幅 [Mbps]	TCP[Mbps]	UDP[Mbps]
1	10	9.33	9.50
2	10 × 2	18.1	18.9
3	10 × 3	26.3	27.7
4	10 × 4	22.5	28.2
5	10 × 5	22.3	26.5
6	10 × 6	22.9	27.4
7	10 × 7	25.3	24.7
8	10 × 8	23.5	27.8
9	10 × 9	23.4	26.3
10	10 × 10	23.1	28.7

ずれがあるため，計算に使用する遅延の値が正確に取れず，上手く振り分けの計算ができなかったからだと考えられる．経路数を多く設定しても，データを伝送していると，最終的に振り分ける経路が 3 経路になってしまうことが多かった．

■TCP，UDP スループット　各経路の遅延を 0.0[sec] として TCP，UDP スループットを求めた．実験では各経路の帯域幅制限を 10[Mbps] に設定し，最大 10 経路までの実験を行った．表 3 は実験結果である．

表 3　TCP，UDP スループット

経路数	帯域幅 [Mbps]	TCP[Mbps]	UDP[Mbps]
1	10	9.33	9.50
2	10 × 2	18.1	18.9
3	10 × 3	26.3	27.7
4	10 × 4	34.9	34.2
5	10 × 5	43.1	46.3
6	10 × 6	50.2	56.4
7	10 × 7	57.6	65.8
8	10 × 8	65.0	75.4
9	10 × 9	72.9	81.0
10	10 × 10	77.8	84.9

実験結果より，複数経路伝送では通信する経路数を増やしていくことで TCP，UDP 共にスループットが高くなることがわかった．経路数を最大 10 経路にすることで，TCP の場合，単一経路の約 8.5 倍，UDP の場合は約 8.9 倍の高スループットを得ることができた．

6　おわりに

本研究では，既存の複数経路伝送システム [2] を使用し，経路数の多いときの複数経路伝送の有効性を示した．そして，このシステムの高速化が期待できるプログラムの改良案を提案した．

今回我々は，複数経路伝送システムを実ネットワーク環境で性能評価することが目的であったが，実験環境が整わず，性能評価実験が行えなかった．しかし，このシステムが実ネットワークで利用が可能であることを確認することができた．

GINE を利用した最大 10 経路での性能評価実験では，経路数を増やすことで TCP と UDP 共に高スループットを得ることがわかり，経路数の多いときの複数経路伝送の有効性を示すことができた．そして，このシステムでは両端ホストで時刻同期を行わないと，振り分けの計算が上手くできず，期待どおりの結果を得ることができないことがわかった．

今後の課題としては，改良した複数経路伝送システムを実ネットワークで使用し，性能評価実験をすることである．プログラムの改良点は，今回提案した Xmit と Relay があり，この部分を改良し，プログラムを作成することで，通信の処理が速くなり，複数経路伝送システムの高速化が期待できると考えられる．この点以外にもプログラムの改良点はあると思うので，システムの最大通信速度を上げることが課題である．他にはシステムを使用することに必要な両端ホストでの時刻同期の機能をプログラムに追加することで，より正確に経路選択をできるようになり，システムの利用価値が高くなると考えられる．

参考文献

[1] 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).

[2] Kawamoto, T. and Goto, K.: Design and Evaluation of IP Multipath Transmission with Feedback, *Proc. of 19th International Conference on Systems Engineering (ICSENG2008)*, pp. 294-299 (2008).

[3] 元井　郁，柳田陽平：複数経路を用いた IP データグラムの配送方法とその性能評価, 卒業論文, 南山大学 数理情報学部 情報通信学科 (2006).

[4] 家田直幸：複数経路を用いた IP パケットの高速伝送方式の提案と試作 , 修士論文，南山大学大学院 数理情報研究科 数理情報専攻 (2006).

[5] 津田尚宏，山田崇詞：複数経路における動画像・音声の伝送とその品質評価, 卒業論文, 南山大学 数理情報学部 情報通信学科 (2008).

[6] 金石朋子，熊崎由佳：複数経路 IP 伝送方式の提案とその上での動画像伝送品質評価, 卒業論文, 南山大学 数理情報学部 情報通信学科 (2007).