平成 14 年度

修士学位論文

パケットアセンブリによるバックボーン網 IP 転送の効率化に関する研究

A study of effecient IP Forwarding for Backbone network by Packet Assembly

1055122　　浦西　慶規

指導教員　　島村　和典

2003 年 2 月 24 日

高知工科大学大学院 工学研究科 基盤工学専攻

情報システム工学コース

要 旨

パケットアセンブリによるバックボーン網 IP 転送の効率化に関する研究

浦西　慶規

インターネットトラヒックの急激な増加に伴い，WDM（波長分割多重）技術が発達しリンク容量の拡大が行なわれている．しかしネットワークは，リンクとノードで構成されており，リンクの広帯域化だけではノードの部分のパケット転送処理がリンク速度に間に合わずボトルネックになる．特にトラヒックの集中するコアネットワークルータでは，IP パケット転送処理負荷が問題になる. そこで，コアネットワークのエッジルータで，行き先が同一である複数の IP パケットを 1 つのパケットにまとめて転送し，コアネットワーク内を流れるパケット数を減少させることにより，コアネットワーク内ルータの IP パケット処理転送負荷軽減を行なう IP パケットアセンブリ転送方式を提案している．本方式では，エッジルータにおいて一定期間パケットを待たさなければならない．この待機時間は，コアルータの負荷軽減とそれぞれのフローの QoS(Quality of Service) パラメータに関係する．待機時間を長くすることは，アセンブリ連結数を増加させ，コアルータの負荷が軽減できるが，反対にそれぞれのパケットへの遅延を大きくし，QoS を低下させてしまう．UDP パケットの遅延は，音声や画像の再生品質に悪影響を与える．また TCP パケットの遅延は，スループットが低下するという悪影響を与える．本論文では，パケットアセンブリによる負荷軽減の方式について述べ，それによるフローへの影響を調査し，改善する提案について述べる．

キーワード　パケットアセンブリ，プロセッサ負荷軽減，アセンブリ待機時間，インターネットトラヒック，エッジルータ，マルチフロー，インターネットエクスチェンジ

Abstract

A study of effecient IP Forwarding for Backbone network by Packet Assembly

Yoshiki Uranishi

Internet link capacity is expanded by WDM(Wave Division Multiplexing) technology, due to the increasing amount of the traffic. However, network consists of nodes and links, so not only link capacity but also high-speed nodes are required.

In this paper, more than one of same IP destination packets are assembled to a packet at Edge Router of Core Network. And, these packets are transmitted efficiently on Core Network Router. Then, they may be divided into the original size at the Edge Router of the opposite. This method is named "Packet Assembly".

This method must make the packets concatenated during a fixed period at Edge Router. This period is relevant to both the packet assembly efficiency and packet delay. Core routers load reduced to set this period for a long time, but the delay of each packet increase. The delay of UDP packet affects the quality of voice communication or video stream. Furthermore, the delay of TCP packet causes decline of throughput.

The purpose of this study is to examine effecient IP Forwarding for Backbone network by Packet Assembly, and investigate its influence to QoS(Quality of Service) of each flow, and improve it.

key words Packet Assembly, Processor Load, IP network, Internet traffic, Edge Router, Multiflow, IX

目次

図目次

表目次

第 1 章

はじめに

この章では，まず研究の目的を述べ，それに至るまでの背景とその問題点を挙げる．様々なネットワークの利用のためにトラヒックの急激な増加が起こっている．それらのトラヒックの集中するバックボーンノードの転送ボトルネックとそれらに対する既存の技術について述べる．

1.1 本研究の目的

本研究は，トラヒックの急激な増加による大容量高速バックボーンネットワークのコアノードのボトルネックを解消するため，インターネットにおける IP パケット転送の効率化を行なうことを目的としている．さらに本方式によるトラヒックへの影響を調査しその課題を発見し効率化を行なうことを目的とする．

1.2 研究の背景と問題点

1.2.1 トラヒック量の急激な増加

インターネットを利用する人口の増加と高速広帯域のネットワークを必要とする様々なネットワークアプリケーションの出現によって，インターネットトラヒックは急激に増加に増加している. 多数のネットワークを相互接続する技術として IX（Internet eXchange）がある．ISP の相互接続を行っている JPIX(図左) と NSPIXP2（図右）の 2 つの IX のトラヒック量の推移を図 1.1 に表している [5][10].

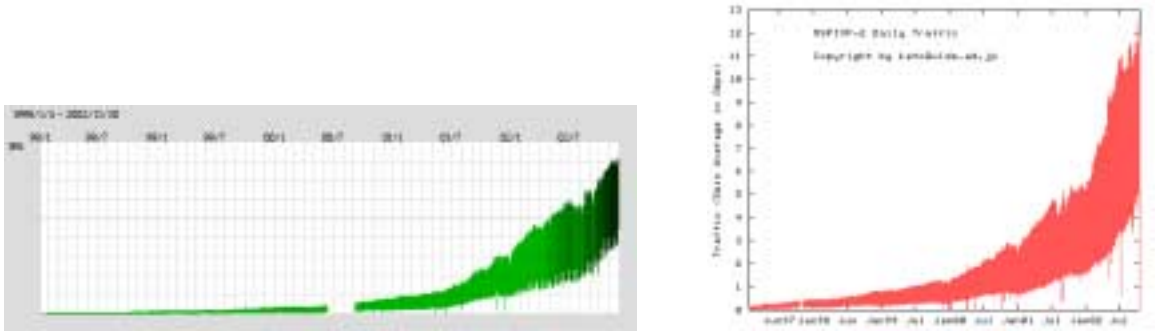

図 1.1　JPIX と NSPIXP2 の IX のトラヒック量の推移（左：JPIX，右：NSPIXP2）

両方の IX とも急激なトラヒック量の増加が見られる．NSPIXP2 の IX サービス開始からトラヒック量は増加し続け，2002 年 1 月から 2002 年 7 月の半年では，トラヒック量が 5Gbps 程度から 10Gbps 程度の約 2 倍という急激な増加が起こっている．

ネットワークアプリケーションとしては，これまでの Web やメールに加えて，一度に複数の大きなサイズの転送を行うファイル共有や，音声通信のように一定の帯域や一定時間内の遅延というような品質を要求するもの，動画のように大容量で高速なネットワークを必要とするマルチメディアアプリケーションの実現の要求が増えることにより，ユーザの増加だけでなくユーザ一人当たりのトラヒック量も増加し，この急激なトラヒック増加は今後も続くと考えられる．

1.2.2　ネットワーク転送のボトルネック

このトラヒックの急激な増加をフォローするように WDM（Wavelength Division Multiplexing）技術が発達し，リンク容量の増加が行なわれている．これまでは，1 本の光ファイバには単波長のレーザー光を通して通信を行っていた．WDM(波長分割多重方式) は, 光の波長を変えることにより, 何本ものレーザー光を多重して一本の光ファイバケーブルに通し, より多くのデータを伝送する技術である．この WDM 技術により，ネットワークのリンク帯域は，Tbit 級または Pbit 級へと広帯域化することが可能になる．

しかし，ネットワークはノードとリンクから構成されておりその両方の高速化が必要とされる．つまりリンクの高速広帯域化だけでは，ノートがボトルネックとなり，大容量高速ネットワークの構築は出来ない．本研究でのノードとは，IP 転送を行うルータのことを示している．

ルータは，リンクから流れてくるパケットのヘッダ処理を行い，適切な次のノードへの転送を行わなければならない．適切なノードへ転送を行うためにルータはルーティングテーブルを作成する．そしてパケットが到着する毎にこのテーブルを参照し，その中から適切な方路を見つけだし，次のノードへの転送を行う．この処理をルーティングというが，現在の急速なネットワークの拡大によりこの検索テーブルの規模が増加しているため，検索に時間がかかってしまう．図 1.2 はルーティングテーブルの経路制御の現状を表したグラフである [6].

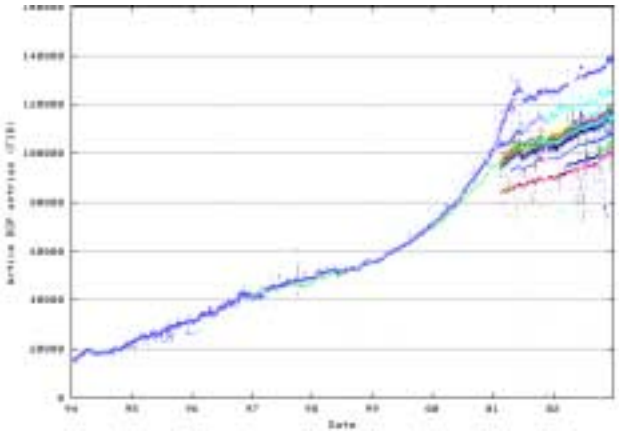

図 1.2　ルーティングテーブルの経路制御情報の現状

2003 年 2 月の時点で，最も規模の大きいテーブルでは，AS1221 の Telstra ネットワークで 140746 件となっている．

パケットが到着するたびにこれらのテーブルから適切な経路を検索する処理負荷が高くなることによって，ノードの転送がボトルネックになることが考えられる．

さらに上記で，トラヒック量の増加は，半年で 2 倍以上である近年の例を説明したが，CPU の処理能力向上の予測として，ムーアの法則では，18 ヶ月で 2 倍であるとされている

ため，これまでの IP ルーティングによる転送方式では，この急激なトラヒックの増加に対応することは難しいと考えられる．特にトラヒックの集中するコアネットワークルータでは，IP ヘッダ処理の負荷がかかり，WDM により多重化されたリンク速度に対応できずにボトルネックになることが考えられる．

1.2.3　バックボーンネットワークノードのボトルネックの詳細

ノードがボトルネックになる原因としてネットワークを流れる小サイズパケットの問題がある．バックボーンネットワークを流れるパケットサイズの分布は図 1.3 のようになっている [7]．1500Byte 以下のパケットが 9 割以上を占めるという結果になっているがこのコア

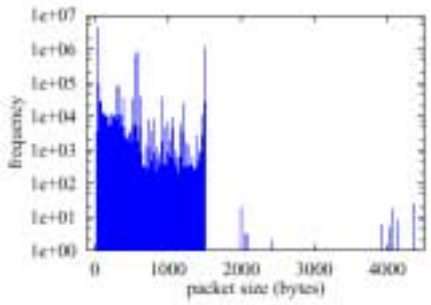

図 1.3　パケットサイズの分布

ネットワークは ATM で構築されており 9180Byte までの IP パケットを転送できる．これによりコアネットワークでは，MTU(最大転送単位) よりはるかに小さいパケットサイズでパケットが転送されていることが分かる．

ここで，MTU とは，ネットワークにおいて 1 パケットで転送可能な最大バイト数である．よって，MTU より大きなデータは，複数のデータに分割して転送される．これは，IP フラグメンテーションと呼ぶ．また，相手先で分割したデータを元のデータに戻すことはデフラグメンテーションと呼ぶ．それぞれのネットワークの特性に応じて MTU の値が異なっている．例えば Ethernet では 1500Byte，ATM では 9180Byte というようになっている．表 1.1 に代表的なネットワークの MTU を示す．

パケットサイズが小さい理由には，以下のようなものがある．

表 1.1　ネットワークの種類と MTU

データリンク	MTU(Bytes)	データリンク	MTU(Bytes)
IP の最大 MTU	65535	Hyperchannel	65535
IP over HIPPI	65280	16Mb IBM Token Ring	17914
IP over ATM	9180	IEEE 802.4 Token Bus	8166
IEEE 802.5 Token Ring	4464	FDDI	4352
Ethernet	1500	Ethernet Jambo Flame	9000
PPP(default)	1500	IEEE 802.3 Ethernet	1492
IP の最小 MTU	68		

- MTU（Maximum Transmission Unit）への適合分割

  MTU の異なるネットワーク同士を接続する場合について考える．MTU の小さいネットワークから MTU の大きいネットワークへパケットを転送する場合，そのままの小さいパケットサイズて転送される．このため，MTU が大きいネットワークでも，他の MTU の小さいネットワークとの接続によって，小さいパケットが流れることになる．また MTU の大きいネットワークから MTU の小さいネットワークへパケットを転送する際にも，Path MTU Discovery（経路 MTU 探索）という，宛先ホストまでパケットを送信したときに分割化（fragmentation）が起こらないようにしてネットワークでのパケットの再構築（defragment）の負荷をなくすため．MTU の小さいネットワークのサイズにあわせたパケットが転送される．

- ロングパケットの転送によるショートパケットへの影響

  帯域の狭いネットワークでは，1 つのパケットの転送にかかる時間が大きくなってしまう．例えば，10Mbps の転送速度のネットワークと 100Mbps のネットワークで Ethernet の最大転送単位での 1 パケット分である 1500Byte のデータ転送の比較を行う場合を考えた場合，それぞれの転送に必要な時間は，

  10Mbps の場合：

$$\frac{1500(Bytes) \times 8(bits)}{10Mbps} = 0.0012s = 1200\mu s$$

100Mbps の場合：

$$\frac{1500(Bytes) \times 8(bits)}{100Mbps} = 0.00012s = 120\mu s$$

となり，低速なネットワークほど 1 つのパケットの転送に時間がかかってしまう．

音声データを IP パケット化する VoIP のパケットは、リアルタイム性を重視するためパケットサイズをなるべく小さく分割して転送される．VoIP トラヒックとロングパケットトラヒックが混在する図 1.4 のようなネットワークの構成で VoIP と FTP(ファイル転送プロトコル) の転送を同時に行った場合を考える．

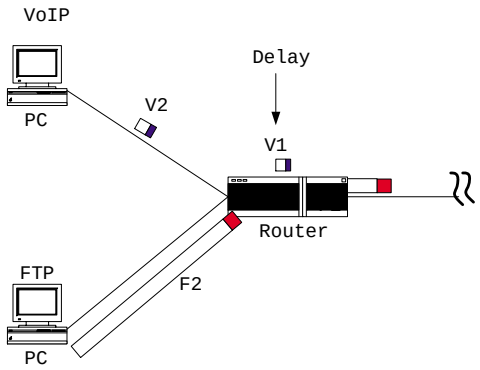

図 1.4　ロングパケットとショートパケットとの混在転送モデル

FTP を行っている端末から転送されるロングサイズの F1 パケットが Router で転送中のため，VoIP のショートパケット V1 の転送が行えなくなる．V1 パケットは次の転送待ちのためにルータのバッファへキューイングされる．この結果として V1 パケットに遅延が与えられてしまう．またインターネットでは複数のノードを経由して転送されるため，それぞれのノード毎に遅延が加算されていくことも考えられるため，大きなサイズのパケットの転送は行うべきではないため小さいパケットサイズで転送されることがある．

- パケットロス率を考慮したパケット転送

  パケットロスは，大きく 2 つに分けることができる．伝送の途中のノードで処理が間に合わずにバッファ等から溢れ出してパケットが破棄される場合と伝送の途中でパケット

の一部分にエラーが発生し，誤ったパケットが相手端末に届いたときに破棄される場合の 2 つの場合が考えられる．ここでパケットサイズが影響するのは後者のほうである．データの伝送には，エラー率が 10^{-9}〜$^{-10}$ 程度である光ファイバ，エラー率 が 10^{-4} 程度である無線等様々なエラー率のネットワークが存在する..

TCP や UDP のトランスポート層によるデータ転送では，パケットのデータの 1 部分でも誤りがあれば，そのパケットのすべてのデータが破棄される．よって，パケットサイズが大きいほどそのパケットが破棄される確立が高くなることになる．このため，エラー率の高いネットワークでは，エラーによる損失を小さくするためにパケットを小さく区切って転送を行う必要がある．

- TCP_Ack パケット

  TCP プロトコルでは，データの転送信頼性を確保するため，届けられたデータパケットに対して，Ack パケットを返す．この Ack パケットは IP ヘッダ＋ TCP ヘッダの小サイズパケットである．データパケットが到着するたびに Ack パケットを返すため，多くの小サイズのパケットが転送されることになる．これは，小サイズのパケットが多く転送されている理由である．

1.2.4　パケットサイズとスループットの関係

ここで，ルータのヘッダ処理負荷の問題として，パケットサイズの問題がある [11].

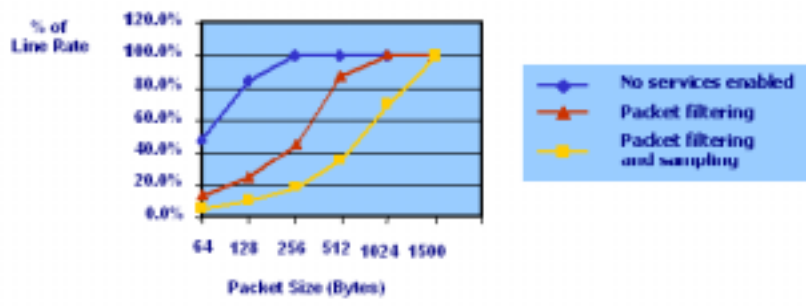

図 1.5　パケットサイズとスループットの関係

縦軸はリンク速度に対する実際のスループット，横軸はパケットサイズをとり，その関係

を表したグラフを以下の図 1.5 に示す．パケットサイズが大きいときには，リンク速度に対してほぼ 100%のスループット出ているが，パケットサイズが小さい場合には，リンク速度に近いスループットが出でないことが分かる．これは，小サイズでルータにパケットが転送される場合，ルータの IP ヘッダ処理の速度が IP 転送に間に合わず，十分なリンク速度が出ていないことが分かる．これは言いかえれば，大きなサイズでパケットを転送し，プロセッサ負荷を下げることにより，リンク速度に近いスループットを実現できることが分かる．

ルータの性能を示す場合には，最小パケットサイズで転送を行った場合の性能を示すことが一般的で，最小サイズのパケットサイズでもリンク速度の 100%を保証しているルータも存在するが，今後，WDM 等のリンク速度の飛躍的な向上や，光ファイバ等でパケットの転送エラーによるロスが小さくなりより大きな MTU が設定されるようになると，図 1.5 と同様に小サイズでのノードボトルネックによるスループット低下の問題が起きると考えられる．

1.3　既存の技術

ソフトウエアルーティングの限界によるノードのボトルネックの問題点を解消する技術として，MPLS，GMPLS がある．また，本研究と同様にパケットをまとめて転送する方式を用いて転送の効率化を行う研究について，問題解決法とその仕組みについて以下に示す．

1.3.1　MPLS，GMPLS

従来のルータの hop-by-hop のソフトウエアベースのパケット転送の限界から，スイッチングの高速化技術として近年注目されている技術として MPLS(Multi Protocol Label Switching) がある [22].

従来の IP による転送では，ルータからルータへと送信され，各ルータがそれぞれ独立して転送先の決定（ルーティング）を行う．そのため，経路上のすべてのルータが IP パケットのヘッダを読み込み，それぞれの宛先を認識し，転送を行う．図 1.6 上．これにより，ルー

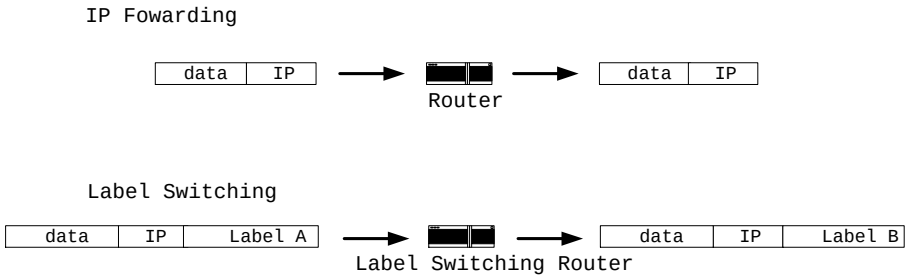

図 1.6 IP ルーティングとラベルスイッチング

タを通過する毎に処理負荷とパケット遅延が発生します.

一方、MPLS では、パケットのラベル付与を MPLS ネットワークの入口（エッジルータ）で行う．パケットは転送される前にラベル付けされ、次のホップにあるラベルスイッチルータ（LSR）は IP パケットのヘッダ解析は行わず、ラベルの情報だけで宛先を認識し、ハードウエアレベルでの高速スイッチングが行える．図 1.6 下.

ここで，ラベルを付与するエッジルータの処理は、従来のルータ処理よりも負荷がかかる．しかし，通常の IP 転送の場合には，エッジルータよりも，トラヒックの集中するコアルータの方が転送にかかる負荷が大きく転送のボトルネックとなる．よって MPLS によるコアルータの転送を行うことにより，IP アドレスよりも負荷の小さいラベル情報を参照して高速な転送を行うことが可能である．これは，エッジルータ負荷とコアルータ負荷のトレードオフを行い効率的な転送を行う方式であため，コアルータの転送処理がボトルネックとなっている現在のネットワークではそのボトルネックを解消する技術として有効である.

また，MPLS は，転送処理の高速化だけでなく，MPLS-VPN や QoS 管理が行える．ラベル番号の管理を行うことにより，あるラベルパスを通過するパケットが別のラベルのパスに入らないようにして MPLS による IP-VPN の構築が行える．例として，企業内の同一グループだけで共通の同じラベルを割り当てるように管理することで，他のラベル番号を持つ企業グループからアクセスできないようすることが可能である.

同様にラベル番号を用いて、ベストエフォート型、高プライオリティ型というようにサービスクラスを割り当て QoS 管理を行うことも可能である.

これらを行うことによりエッジルータにかかる負荷は大きくなることが考えられるため，ラベル付与の効率化とコアルータの負荷とのバランスが重要であると考えられる．

1.3.2　他の研究

本研究以外にも，小サイズのパケット転送の問題点を挙げ，パケットをまとめて転送し，ルータの負荷を軽減することを目的としている研究がある [12][13]．これらはルータのバッファ（FIFO）からパケットを取り出す際に，バッファに格納している連続する他のパケットの宛先を参照し，方路が同じパケットは多重化して送信する方式である．

この方式では，通常のエッジルータにかかる負荷によってバッファに蓄積されたパケット同士で連結を行っており，後続のパケットを待ち合わせるようなことはしていなため，パケットをまとめることによる遅延は起こらないと考えられる．よって本研究では，エッジルータの負荷はパケットをまとめることにより上昇することが考えられているが、これらの研究では，パケットをまとめて転送することにより，連結を行うノードにおいても通常の転送より低い負荷で転送が行えることをシミュレーションで示している．

この実験では，エッジルータの負荷が 100 %を超える状態での実験を行っている．しかし，エッジルータの負荷はその数を増やすことにより軽減できると考えられ，エッジルータの負荷が 100 %を超える状態であることは，ほとんどないと考えられる．よって，エッジルータの負荷が小さい場合にはパケットが蓄積されず，パケットをまとめることができないと考えられる．また，エッジルータの負荷が低くても，トラヒックの集中するコアルータの負荷が高い場合は考えられるため，エッジルータの低負荷時にはパケットをまとめることができないという問題が考えられる．また，これらの研究では，フロー毎にパケットをキューイングしないため，宛先が同一であるパケットが連続的に到着した場合にしかパケットをまとめることができないため，トラヒックが混在するバックボーンネットワークでの適用は難しいと考えられる．

.

第 2 章

パケットアセンブリの説明とその検討

パケットアセンブリとは, コアネットワークへパケットを転送する際に, エッジノードで複数の宛先が同一であるパケットを一つのパケットにまとめて転送する方式である. これによりトラヒックの集中するコアノードで,IP ヘッダ処理回数の削減が可能となりボトルネックの問題を解決できると考えている.

この章では，パケットアセンブリについての説明を行いそれによる効果について述べた後，パケットアセンブリをネットワークにどのように適応するかについて述べる．さらに本方式の課題と効率化について述べる.

2.1 パケットアセンブリについて

図 2.1 のようにアクセスネットワークの nodeA からパケットが転送され，コアネットワークを経由して反対側のアクセスネットワークの nodeB へパケットを転送する場合を考える.

nodeA から転送されたパケットは，コアネットワークの Edge Router1 で宛先が同一である複数のパケットを一つにまとめる．インターネットでは，様々な宛先のパケットが転送されているため，それらのパケットを分類する必要がある.

宛先毎にまとめられたパケットは，図 2.1 のように一つの大きなサイズの IP としてコアネットワーク内のルータに転送される．これにより，コアネットワーク内のルータでは，IP

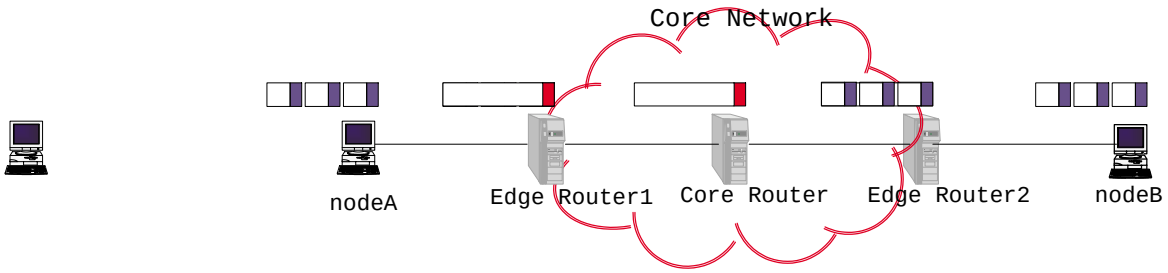

図 2.1　パケットアセンブリ網

ヘッダ処理の回数が軽減される．この例では 3 つのパケットを一つに纏めているためコアネットワークのルータでは通常の 3 分の 1 の IP ヘッダ処理負荷で転送が出来ることになる．

コアネットワーク内のルータで効率良く転送されたあと，反対側の Edge Router2 で再びもとのパケットに分割して反対側の nodeB へ転送する．この一連の転送方式をパケットアセンブリと呼ぶ．

図 2.2 に IP ヘッダのフォーマットを示す．

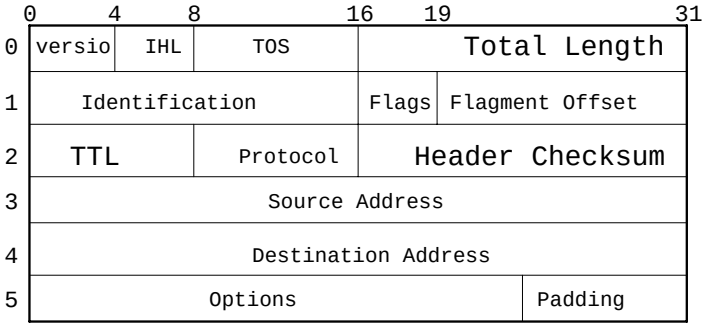

図 2.2　IP ヘッダフォーマット

コアルータでは，アセンブリされたパケットは通常の IP パケットとして扱われるので，コアルータが特別なアセンブリのための機能を実装する必要はない．このためパケットアセンブリ網のエッジルータだけにこの方式を用いることで負荷軽減を行えるという特徴を持っている．

アセンブリされたパケットは，ルーティングを行う分には通常のパケットと変わらないが，そのパケットを受信するノードにとっては，IP のペイロード部分が通常のパケットと

はことなるため受信処理が正確に行えなくなる．よって，受信を行うためには，必ず反対側のパケットアセンブリ装置でディスアセンブリ（パケットを元に戻すこと）を行ってからそれぞれのノードへ転送を行わなければならない．バックボーン内のノードへのパケット転送を行う場合には，そのノードにディスアセンブリ機能を持たせるか，その宛先へのパケットのアセンブリを禁止するような仕組みを用いて，アセンブリされたパケットを処理してしまわないように工夫する必要がある．

また IP ルーティングでは，ルータを経由する毎に TTL の減算か行われるが，アセンブリされた後続のパケットはデータとして扱われるため TTL の減算を行えない．よってディスアセンブリ時に，先頭の IP ヘッダの情報から後続のパケットの TTL の減算を行う必要がある．

2.2　パケットアセンブリ方式の課題

2.2.1　アセンブリ待機時間による遅延とその影響

ルータは，処理能力を超える量のパケットが到着した時には，バッファを利用して一度メモリに蓄積し，到着順に処理していく．しかし，処理する量が少なく，到着したパケットが次のノードへ転送可能であれば, そのまま転送される．

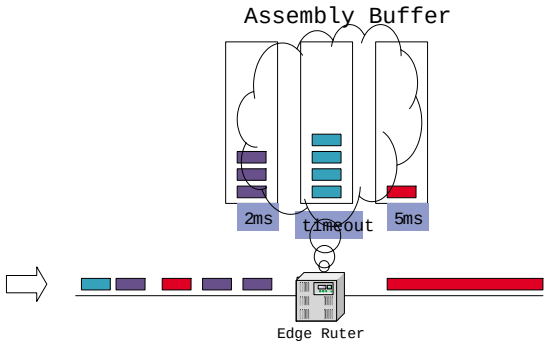

図 2.3　アセンブリバッファ

パケットアセンブリでは, 複数のパケットをまとめるために負荷がかかっていない場合で

も，エッジノードに到着したパケットは, 図 2.3 のように一定の期間，後続のパケットを待たなければならない. これをアセンブリパケット待機時間と呼ぶ．これにより待機しているそれぞれのパケットには, アセンブリのための遅延が生じる.

アセンブリパケット待機時間を長くするほど，より多くのパケットを 1 つにまとめることが可能であるが，それは個々のパケットの遅延を大きくし，データ転送に悪影響を与えてしまう．この悪影響については，帯域が 100Mbps，1Gbps と広帯域になるに従い，大きなサイズの転送にかかる時間が小さくなっていくことにより，問題は無くなっていくことを [17] で示している.

また，遅延の問題として，アセンブリの後続パケットの待ち時間ための遅延が生じるため，実時間通信の Delay センシティブなパケットに対しては，アセンブリを行なわずそのまま転送することで，Delay の悪影響を無くすことを考えている．また Delay ロバストな非優先パケットに対しては，その非優先による待ち時間を利用してアセンブリを行うことを本研究では提案（Priority Queuing with Assembly）している．QoS を実現するために使用される通常の Priority Queuing では，優先キューと非優先キューを用意する．そして，アプリケーションの特性に応じてそれぞれのパケットを分け，優先キューに蓄積されるパケットを優先的に転送し，Delay センシティブなパケットの Delay を小さくすることを目的としている．Priority Queuing with Assembly は，この機能に加え非優先キューにパケットアセンブリ機能を追加した方式である．この方式のシミュレーションを行うことでそれらの影響について検証している [3].

さらに，遅延の影響を受けるパケットとして TCP の確認応答に使用する Ack パケットがある．このパケットへの遅延が閾値以上であると，送信したデータが，端末に到着しているにも関わらず，TCP タイマのタイムアウトにより到着していないと判定され，データの再送が行なわれてしまう.

TCP パケットの遅延のもうひとつの影響として，TCP スループットの低下が考えられる.

TCP の転送のウインドウ制御による経過図 2.4 にを示す．（A）は RTT(往復伝播遅延)

が大きい場合，（B）は RTT が小さい場合の通信の様子を示している．RTT が小さい場合

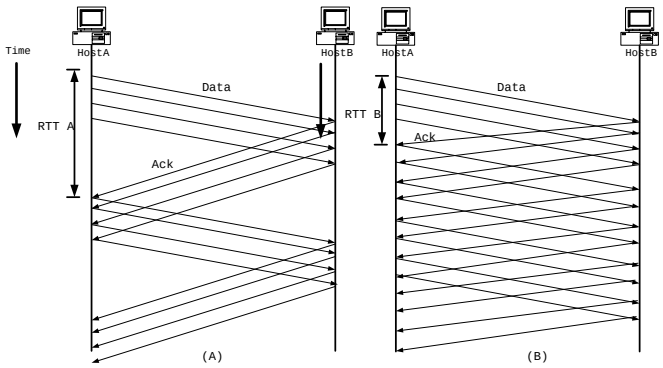

図 2.4　TCP ウインドウ制御

は，同じ時間内により多くのデータが転送できることが分かる．これは，TCP では一度に転送できるデータ量が決まっておりそのデータの Ack が返ってくるまで，次のデータの転送は行えないためである．

以下に TCP のスループットと RTT の関係を示す．

$$Throghput \propto \frac{Windowsize}{RTT}$$

この式から，スループットは RTT に反比例することが分かる．スループットは，bps（bit per second）で表し，1 秒間に処理したデータ量を示す値である．

しかし，実際の通信では，相手の端末に届くまでの時間は 1 秒よりも小さい ms の単位である．つまり，この一度に送る動作を 1 秒間に何回行えるかが TCP のスループットに影響する．例えば，RTT が 10ms と 50ms の場合で比較を行うと，10ms の場合は，1 秒間に

$$\frac{1s}{10ms} = 100 \text{ 回}$$

となり，1 秒間でウインドウサイズ分の転送を 100 回行えることになり，50ms の場合には

$$\frac{1s}{50ms} = 20 \text{ 回}$$

となり，1 秒間でウインドウサイズ分の転送を 20 回行えることになる．従って，RTT が大きくなればなるほどウインドウサイズ分の転送が行える回数が減ることから，RTT が TCP スループットに影響を与えることが分かる．

アセンブリによってスループットが低下したことによってコアルータの負荷が低下することは，1 つのフローの転送時間を長時間にするだけであり効率の良い転送が行えるとは言えない．そこで，アセンブリを行うために必要な後続のパケットを待機するアセンブリタイムアウト時間を短く，またコアルータの負荷を軽減するために多くのパケットをまとめることが重要である．

2.3　MTU（最大転送単位）による連結の制限

パケットアセンブリを行う場合，パケットの待機時間による連結の制限ではなく，MTU による制限が使用される．例えばコアネットワークの MTU が 9000Byte であり，アクセスネットワークから，1500Byte のデータが 7 パケット転送されそれらをまとめる場合を考える．このとき 1500Byte × 7 パケット＝ 10500Byte となり MTU の制限を超えてしまうためこのパケットは破棄されてしまう．そのため，MTU の制限を越える場合には連結を終了しなければならない．また，MTU による制限が適用されるフローは，パケット待機時間分の遅延が小さくなるため高速な転送が行えることになる．それに対して，パケット待機時間の制限で転送されているフローは，常に遅延が大きくなり高速な転送が行えない．それぞれのフローがこの 2 つの制限のどちらに適用されるかで，意図的でない転送速度の差が出てしまうことが考えられる．

2.4　パケットアセンブリの効率化

アセンブリタイムアウトの時間を短くしフローへの影響を小さく，より多くのパケットをまとめる方式としてマルチフローアセンブリを提案している．ここでは，このマルチフローアセンブリの仕組みとそれにより期待される効果について説明する．

2.4.1　マルチフローアセンブリ

マルチフローアセンブリとは，複数の行き先が同一であるフローを一つのフローとして扱うアセンブリ方式である．

マルチフローアセンブリの概要を図 2.5 を用いて説明する．端末 A1，A2，A3 からアセンブリエッジルータ R1，コアルータ R2，エッジルータ R3，R4 を経由してそれぞれの端末 B1，C1，C2 へパケットを転送するフロー F1，F2，F3 を表している．

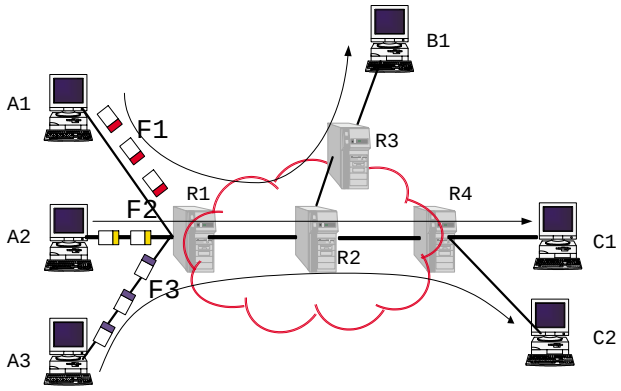

図 2.5　マルチフローアセンブリ

端末 A1 から流れるフロー F1 は，R1 でアセンブリされた後，エッジルータ R3 で再び元のパケットへもどされた後，端末 B1 へ流れており通常のパケットアセンブリが行われる．次に端末 A2，A3 から転送されるフロー F2 と F3 について，これらの 2 つのフローは，R1 でアセンブリされ，反対側の R4 で元のパケットにもどされるという同じ経路をたどり，そのあとでそれぞれの端末 C1，C2 へ流れており通常のパケットアセンブリでは，F1 と同様にアセンブリされる．

ここで，マルチフローアセンブリでは，これらの 2 つのフローのパケットをまとめて一つのフローとして扱い，アセンブリを行う．これにより，アセンブリを行う R1 では，まとめられたフローのパケットの到着が増え，同じアセンブリタイムアウト値でより多くのパケットをまとめることができると考えられる．またアセンブリ連結数でアセンブリの制限を考えた場合には，通常のアセンブリ時より短い時間でアセンブリパケットを転送することが可能

になると考えられる．

2.4.2　マルチフローアセンブリのバッファ

次にマルチフローアセンブリのバッファについて説明する．通常のパケットアセンブリバッファとマルチフローアセンブリのバッファを図 2.6 に示す．図右が通常のパケットアセンブリのバッファ，図左がマルチフローアセンブリバッファを図示している．

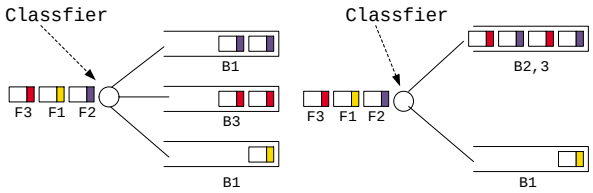

図 2.6　アセンブリのバッファ

通常のマルチフローアセンブリを行うアセンブリバッファでは，F1，F2，F2 のそれぞれのフローのパケットが B1，B2，B3 にバッファされ，別々にパケットアセンブリを行う．マルチフローアセンブリバッファでは，F2、F3 のフローのパケットは同じバッファ B2,3 に蓄積される．このため，B2,3 では，同じアセンブリ時間内により多くのパケットが蓄積できるため、効率の良いパケットアセンブリが行えること考えられる．

第 3 章

実ネットワーク（高知工科大学 Network）トラヒック調査

この章では，実際にインターネットへ流れていくトラヒックを計測，分析した結果から，アセンブリ方式の検討を行う．トラヒックの取得方法について説明した後，トラヒックデータの概要について述べ，パケットアセンブリに関する分析結果を考察する．

3.1 トラヒックデータの取得

トラヒックデータの取得は，パケットの取得を行いたいノード間のトラヒックを，パケットキャプチャソフトを用いてパケットの一つ一つのデータを記録することにより行う．ノー

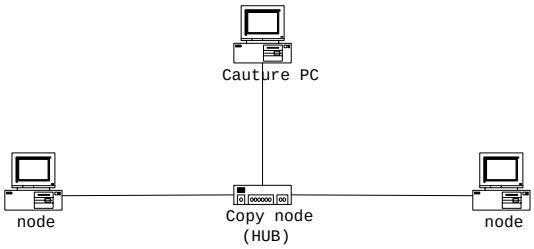

図 3.1 パケットキャプチャ

ド間の通信を切断しないようにするため，その間に回線の接続されたすべてのポートにパケットをコピーして転送する HUB や特定のポートのみにパケットをコピーして転送するミ

ラーリング機能を持ったノードを用いることで，図 3.1 のように，パケットのキャプチャを行う．

パケットキャプチャソフトは,Ethernet アダプタをプロミスカス・モードという自分の宛先以外のパケットも受信する特殊な動作モードにして利用している. 通常、Ethernet アダプタはネットワークケーブルから受信した通信パケットの Ethernet ヘッダを調べ, 宛先 MAC アドレスを比較し, ブロードキャストまたは自分の宛先と一致するパケット以外は, 破棄するように動作する.

一方, プロミスカス・モードでは, この宛先 MAC アドレスのチェックを行なわず, 受信したすべての通信パケットを上位の階層に引き渡すようになる. パケットキャプチャソフトは, このような仕組みを利用して, ドメイン内のすべての通信パケットをキャプチャして解析することを可能にしている.

3.1.1　トラヒック測定ポイント

インターネットトラヒックデータを取得するために, 図 3.2 のように高知工科大学（KUT Network）とインターネットの境界部分で測定ツールによりパケットキャプチャをおこなった。測定ポイントのネットワークは，ARP パケットや IPX 等のパケットも転送されているが今回は IP パケットのみの取得を行いそれ以外のパケットはキャプチャしないようにした.

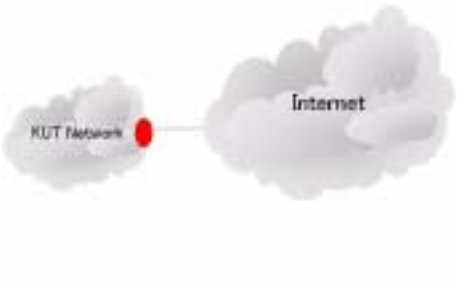

図 3.2　測定ポイント

3.1.2　トラヒック測定ツールについて

パケットをキャプチャする既存のツールとして，Sniffer や Ethereal などがありそれらを用いればキャプチャを行うことが可能である．

しかし，いずれのツールも今回では必要のない情報までキャプチャするため，長時間の測定はハードディスクに記憶できても，メモリ不足によりそれらのファイルを展開することはできないことが分かった．またこれらのデータはテキストデータではないため，取得ファイルの分割を行うことも出来なかった．

そこで，必要なデータのみの取得を行い，テキスト形式で保存を行うプログラムを新たに作成し，それを用いてパケットキャプチャを行った [18]．

3.1.3　測定データフォーマット

次に測定ツールの取得データについて説明する．

- パケット No：パケットの到着順に番号を付与する
- source address：パケットの送信元 IP アドレス
- destination address：パケットの送信先 IP アドレス
- パケットサイズ：パケットのサイズ（物理層のヘッダも含む）
- ホップ数：IP ヘッダのホップ数
- プロトコル番号：IP ヘッダのプロトコル（上位プロトコル）
- source port：TCP または送信元 UDP のポートナンバー
- destination port：TCP または UDP の送信先ポートナンバー
- シーケンスナンバー：TCP パケットのシーケンスナンバー（TCP 以外は 0）
- Ack ナンバー：TCP パケットの Ack ナンバー（TCP 以外は 0）
- 秒：パケットが到着した時間（秒）1970 年 1 月 1 日 0 時からの経過時間（秒部分）
- μ秒：パケットが到着した時間（μ秒）1970 年 1 月 1 日 0 時からの経過時間（μ秒（

10^{-6}

秒）部分）

実際にはテキストに以下の例ような形式でパケットのデータを記録していく．

0 210.163.144.223 210.150.208.125 66 61 6 32310 80 3449075585 557267221 1003711713 646659

1 210.163.148.169 63.79.239.169 60 123 6 61200 8888 12977276 4151853812 1003711713 649403

2 210.163.148.169 198.172.95.235 60 123 6 62689 8888 29652262 296870322 1003711713 650148

3 210.163.148.169 66.92.171.175 60 123 6 62592 8888 14542443 1453718538 1003711713 650936

4 210.163.148.169 66.52.8.32 60 123 6 64478 8888 12284757 3469707398 1003711713 651918

1 番目のパケットを例としてあげると，パケット No が 0，送信元アドレスが 210.163.144.223，送信先アドレスが 210.150.208.125，パケットサイズは 66Bytes，ホップ数 61(残りのポップ数)，プロトコル番号 6(TCP)，送信元ポート番号 32310，送信先ポート番号 80(HTTP)，シーケンスナンバー 3449075585，Ack ナンバー 557267221，到着時間 1003711713.646659 秒というデータが得られる．

3.2　取得トラヒックデータの概要

取得したトラヒックデータの概要を表 3.1 に示す．

表 3.1　トラヒックのデータ

No	測定開始日時	時間	パケット数	転送量
1	2002.1.24 9:00	33h	34,753,032	7.962 GBytye
2	2002.1.25 18:00	63h	56,602,119	13.076 GBytye
3	2002.1.28 9:00	48h	52,058,353	11.58 GBytye
4	2002.1.30 9:00	24h	27,557,640	6.577 GBytye

取得したデータは，2002 年 1 月 24 日 9:00 から 2002 年 2 月 31 日 9:00 までの 1 週間分のトラヒックデータである．IP パケット総数は 170,971,037 パケットであった．また総バイト数は 39GBytes であった．表では，4 つのファイルに分かれているが，これは測定時に

一週間分のトラヒックでハードディスクの容量を越えないように，データの圧縮を行い，測定を分割したためである．

3.3　トラヒックデータの分析

この章では，トラヒックデータの分析を行った結果について述べる．またパケットサイズの分布以降で扱うトラヒックデータは，アセンブリを行うパケットを対象にするため，高知工科大学からインターネットに向かうパケットのみを扱っている．

3.3.1　各プロトコルの出現頻度

プロトコルについて分析を行い出現頻度の高い上位３番目までを表した分析結果を表 3.2 に示す．

表 3.2　プロトコル別

	各パケットの割合（%）
TCP	99.168
UDP	0.738
ICMP	0.092

TCP プロトコルが 99.168 %と圧倒的に多いことが分かる．よって，このネットワークでのパケットアセンブリでは，TCP パケットに着目したアセンブリを行うことが最も影響が大きいことが分かる．また，UDP パケットなどの Delay センシティブなパケットは，アセンブリを行わずにそのまま転送するようにしても，その分の負荷軽減に対する影響はほとんどないと考えられる．

しかし，今後 VoIP や映像のストリーミング等の Delay センシティブなパケットを用いるアプリケーションが増加することが考えられる．データ系アプリケーショントラヒックとの割合によっては，UDP パケットのアセンブリを考える必要もある．

3.3.2　パケットサイズの分布

パケットサイズの分布を図 3.3 に示す.

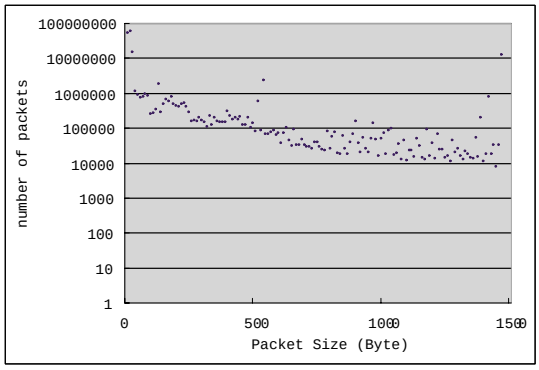

図 3.3　測定した総パケットのサイズ分布

MCI トラヒック（図 1.3）のパケットサイズ分布の結果と同様に，60Byte 付近の小サイズパケットが多く存在していることがわかった．測定を行ったデータリンク層は，Ethernet であるため MTU の 1500Byte 以下のパケットが転送されている．これらのパケットはこの先の ATM ネットワークを経由することになるが，ATM の MTU は 9180Byte であり，MTU より遥かに小さいパケットが転送されているということが分かる．

次に，パケットアセンブリでは，アクセスネットワークからコアネットワークへ出ていくパケットをアセンブリするので，高知工科大学からインターネットへ出ていく方向へ流れていく outgoing パケットのみのパケットサイズ分布を図 3.4 に示す．70Byte 以下のパケットが半数以上であり，プロトコル別の出現頻度で TCP の割合が 99.168%であったことから，TCP Ack パケットの占める割合が大きいと考えられる．この Ack パケットをアセンブリすることは，コアネットワークルータの負荷軽減に重要である．

アセンブリされた Ack パケットがロスした場合には，多くの再送が起こることが考えられるが，アセンブリを行うコアネットワークは，アクセスネットワークに比べ低ロスのネットワークを想定している．またロスした場合でも，Ack の確認の仕組み上（Ack パケットはどこまでのデータが到着したかを相手端末へ知らせる），TCP タイムアウトが起こり再送さ

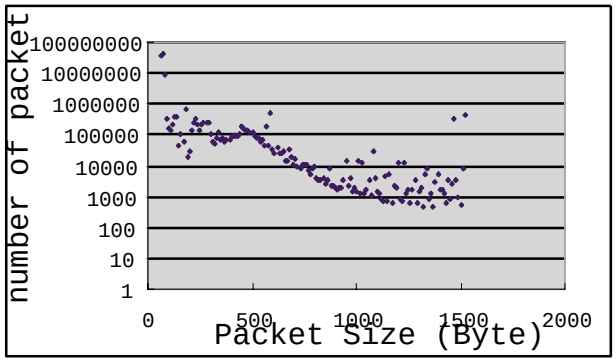

図 3.4　測定した Outgoing パケットのサイズ分布

れるまでに，それ以降の Ack パケットが到着すれば，ロスした Ack パケットの確認を含む応答が返ってくることになるので，問題はないと考えられる．しかし Ack パケットが遅れることによるフローへの影響は考えなければならない問題点である．

3.4　パケット到着間隔の分析

パケットアセンブリでは，まとめるパケットの到着間隔が小さければ小さいほど短い時間でアセンブリが行えるため効率的である．ここでは測定したトラヒックをフロー毎に分けそれらの到着間隔を分析する．トラヒックの分析の手順を図 3.5 に示す．

図中の 1. は取得したすべてのパケット（No.1〜No.n）である．これらのパケットを No.1 のパケットから順に 2. のように宛先 IP アドレス別に分ける．その後，それぞれのフロー毎に 3.（矢印はパケット到着間隔の時間を示す）のようにパケット間隔を求める．

このパケット到着間隔を用いて以下のようなトラヒック分析を行った．またこのフロー分析には 100 万 packet（表 3.1 の No.2 の約 2 時間分）のトラヒックを用いた．

3.4.1　アセンブリタイムアウトとコアネットワーク内パケット数

取得したパケット到着間隔をもとに，実際にパケットアセンブリを適応したときのタイムアウト値とその時のコアネットワークへ転送されるパケット数の変化について分析する．各

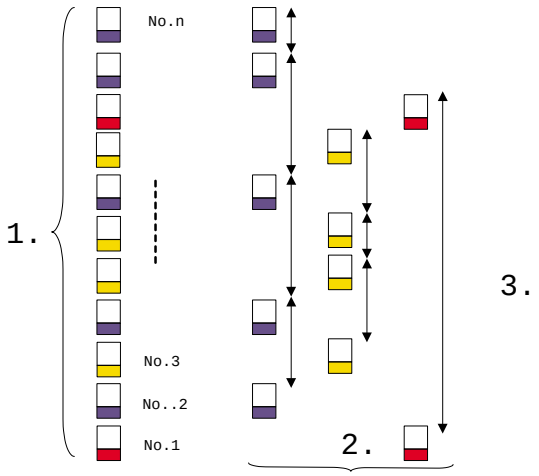

図 3.5　フロー毎のパケット到着間隔

フローの先頭パケットから順に到着間隔を加算していき，アセンブリタイムアウト時間以内であるパケットを 1 つにしたときの全体のパケット数のカウントを行った.

それぞれのタイムアウト値にたいするコアネットワーク内パケット数を図 3.6 に示している. 縦軸はコアネットワークのパケット数, 横軸はタイムアウト値を示している. タイムアウ

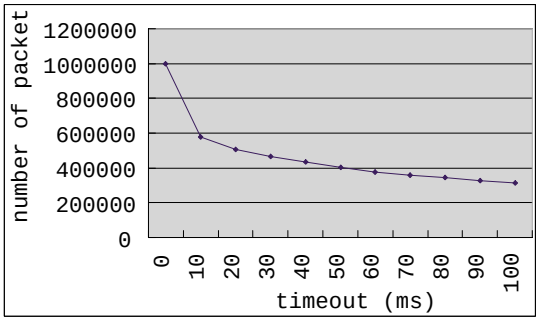

図 3.6　アセンブリタイムアウトとコアネットワーク内パケット数

ト 0ms（パケットを待機させない）の場合の, コアネットワークへ流れるパケット数は,100 万パケットである. このときタイムアウトを 10ms,20ms,…と増加させるに従い, コアネットワークを流れるパケット数が減少していくことがわかる. タイムアウトが 20ms には, コアネットワークを流れるパケット数は通常の約半分のパケットで転送されることになる. またさらにタイムアウト値を大きくした 100ms の時には，コアネットワークへのパケット数は

通常の 3 分の 1 以下の転送になることになる．

工科大とインターネットの接続ポイントの帯域は上り、下りともに 6Mbps であった．

3.4.2　アセンブリタイムアウトの 2 つの方式

パケットを連結するために，エッジルータでパケットが次のノードへ転送可能状態であっても，ある一定期間パケットを待機させて，その間に到着した行き先が同一である複数のパケットをまとめて転送する．この待機時間の制限をアセンブリタイムアウトと呼ぶ．このアセンブリタイムアウトについて，２つの方式を図 3.7 のように検討している．図の矢印はアセンブリタイムアウト（パケット待機時間）の長さを表している．

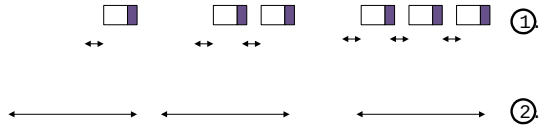

図 3.7　アセンブリタイムアウトの 2 つの方式

方式　は，アセンブリを行うフローのパケット間隔がタイマ値よりも超えた場合に，アセンブリのための後続パケットの待機を停止し，アセンブリパケットを次のノードへ転送する方式である．タイマはパケットが到着するたびにリセットされる．

利点：連続的にパケットが到着しなければ，即パケットを転送することが可能なため，全体的なパケット待機時間が短くなる．

欠点：パケットがタイムアウト以内に到着すれば，タイマはリセットされ続けるので，サイズの小さいパケットが，アセンブリタイムアウト以内に次々に到着した場合は，MTU，タイマのどちらの制限にもかからず，大きな遅延を与えてしまい最大遅延時間が定まらず，遅延の揺らぎを生成させることが考えられる．特に VoIP 等の小サイズで，一定間隔でのパケット到着があり，Delay センシティブなパケットのフローに対しては，大きな遅延と揺らぎを与えてしまう．

またパケットサイズによって遅延時間が異なるため，アセンブリモジュール内部での最大

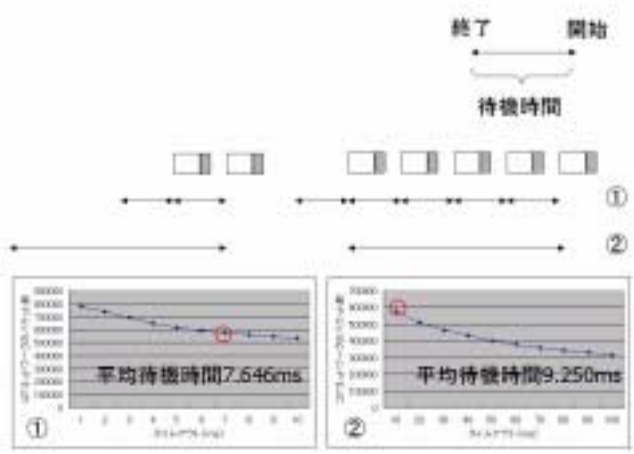

図 3.8　2 つのタイムアウト方式の比較

遅延時間が保証できないことも問題である．

方式　は，アセンブリ開始時に設定されたタイマを起動し，タイムアウトになった場合に，アセンブリのための後続パケットの待機を停止し，アセンブリパケットを次のノードへ転送する方式である．パケットの到着によるタイマのリセットは無く，アセンブリを行う最初のパケットのみタイマのセットが行われる．

利点：先頭のパケットが到着してから，タイムアウト以内に到着したパケットでアセンブリを行うため，タイムアウト値以上の遅延を与えることが無く，アセンブリモジュール内部での最大遅延時間が保証できる．

欠点：パケットの到着間隔に関係無くすべてのパケットに対して，パケットをまとめるだけのアセンブリタイムアウト時間だけ待機させられるため，到着間隔が広く，アセンブリできないフローには遅延が加わるだけの効率の悪い転送になってしまう．

これらの特徴から考えると，パケット間隔が狭くバースト的にパケットが到着し，かつ Delay ロバストなアプリケーションでは，　の方式を用いることが効率的であると考えられ，また Delay センシティブなアプリケーションでは，アセンブリ最大遅延時間が保証できる　の方式が有効であるとと考えられる．

グラフは，各タイムアウト方式による，コアネットワークパケット数の軽減を表したもの

であるが，両方式でパケット数を 100 万パケットから 60 万パケットまで減少させた場合のそれぞれの平均パケット待機時間を計算すると，方式　では，7.646ms，方式　では 9.250ms となった．この結果からは，方式 1 の方が待機時間が少なく効率の良いアセンブリが行えることが分かる．

3.4.3　フロー毎のパケット到着間隔の違い

測定したトラヒックは，様々なアプリケーションから発生するフローが存在する．アプリケーションの違いによりそれらのフローのパケット到着間隔も異なると考えられる．またアプリケーションの違いだけでなく，接続先のネットワークの帯域などによってもフローのパケット到着間隔は異なる．このフローごとのパケット到着間隔の違いを調べる．

ここでは，トラヒックをフロー別に見た時に，フローの大きい順（パケット数が多い）に 100 フローを利用した．これらフローはコアネットワークへの影響を大きく与えるもとのして取り出した．これらのフローのパケット到着間隔がタイムアウト以内である割合を求めた．

パケット到着間隔がタイムアウト時間以内であるという割合 X（%）は，あるフローで Y 個のパケット到着間隔が存在し，それらのパケット到着間隔がタイムアウト時間以内であるパケット到着間隔をカウントした数を Z 個とすると，

$$X = \frac{Z}{Y} \times 100$$

で求められる．

横軸がアセンブリタイムアウト時間を示し，縦軸がフロー毎のパケット到着間隔がタイムアウト時間以内である割合を図 3.9 に示している．タイムアウトが 10ms でパケット到着間隔がそれ以内である割合が 80%程度であるものから，タイムアウトをいくら大きくしてもその割合が増加しないものまでさまざまなフローが存在することがわかる．

このことから考えられるアセンブリ方法について以下に考察する．

- フロー毎に異なるタイムアウトを設定する

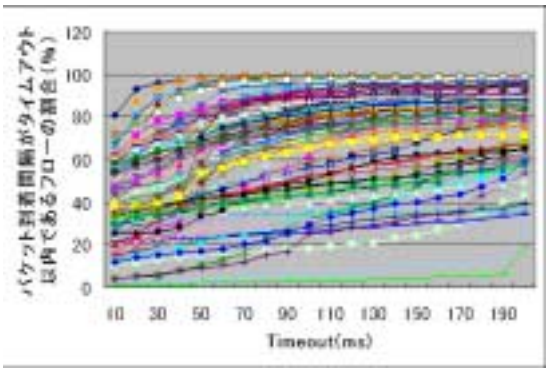

図 3.9　フローのパケット到着間隔の違い

フロー毎にパケット到着間隔が異なるのであれば，フロー毎に異なるタイムアウトの設定値を用いることが考えられる．これにより，それぞれのフローに適したアセンブリが行え無駄な遅延が起こらず効率の良い転送が可能となる．また，同一のフローであってもネットワークの状態によってパケット到着間隔が異なることも考えられるためタイムアウト値を変動させることも考えられる．しかし，これら場合，パケット到着間隔がの違いをどのように区別し適切なタイムアウト値を求めるかのタイムアウト設定方法やその処理のために必要なフローの管理負荷が問題になると考えられる．

- 全体のトラヒックで共通のタイムアウトを設定する

  フロー毎に設定を行わずに共通の設定を用いるのであれば，ネットワークにパケットアセンブリを導入する際に全体のトラヒックを分析し，最適なアセンブリタイムアウトを求めることで全体として効率的なアセンブリを行うことが考えられる．また，すべてのフローに共通のタイムアウト値ではなく，アプリケーション別やネットワークアドレス別等のフローの分類を行いそれらで共通の設定を用いることにより，よりフローに適した効率的なアセンブリが行え，フロー毎にタイムアウト値を設定するよりも小さい負荷で行えると考えられる．

- 到着間隔の異なるフローを混在させる

  本研究では，マルチフローアセンブリを提案しているが，パケット到着間隔が狭いフローといっしょにパケット到着間隔が広いパケットをマルチフローアセンブリすること

で，パケット到着間隔が広く余りアセンブリできないフローでも，効率的にアセンブリを行うことが可能であると考えられる．

3.4.4　アドレスの集約

宛先 IP アドレスを用いてマルチフローアセンブリを行うために重要であると考えられるアドレスの集約についての調査を行った．測定したパケット（サンプルの 100 万パケット）をアドレスのビットを変化させ，そのビットまでのアドレスが一致するものは同一のフローのパケットとして扱った場合のフロー数を測定した結果を表 3.3 に示す．

表 3.3　マルチフローによるフローの集約

一致させるアドレスのビット数	フロー数
32bit(172.21.33.51)	6552
24bit(172.21.33.)	3365
16bit (172.21.)	198

一致させるビットを 32 から 16 へと小さくしていくとフロー数が減り，フローの集約が行えることが分かる．アセンブリではフローの数だけバッファを用意しフロー毎にパケットの蓄積を行うため，アドレスの集約によりバッファ数を削減できる．アドレスの完全一致でアセンブリを行った場合には 6552 フローであったものが，16bit の一致では 198 フローにまで集約されている．よってマルチフローアセンブリは，アセンブリエッジルータのバッファ管理負荷軽減に有効であると言える．

マルチフローアセンブリの効果としてフローのパケットの到着率を上げることで効率的にアセンブリを行うことを考えている．しかしこの実験でのフローの集約は 100 万パケット中，約 2 時間分という長い時間で見た場合に行える集約であるため，アセンブリタイムアウトのように短い時間内にフローの集約が行えるとは言えないため，実際にこのようなアドレスの集約を行った場合のパケットアセンブリの検証を行うべきである．

3.4.5　マルチフローを用いたときのアセンブリタイムアウトとコアネットワーク内パケット数

マルチフローアセンブリを行った場合のフロー集約とその効果を検証する．アセンブリタイムアウトとコアネットワーク内パケット数の関係を通常のパケットアセンブリ（シングルフローアセンブリ）とマルチフローアセンブリを用いた場合とで比較を行った．また，マルチフローのアドレスの集約は 16 ビットまでの一致としている．

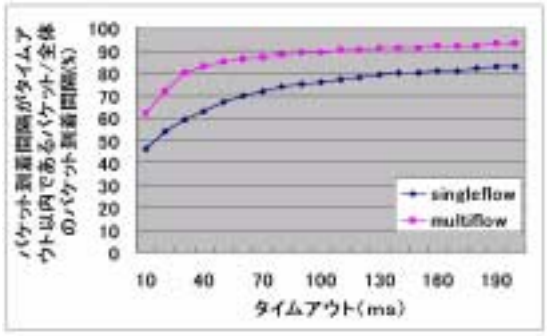

図 3.10　マルチフローの実験

マルチフローでアセンブリを行った場合には，シングルフローでアセンブリを行うより多くのパケットをまとめることができることが図 3.10 で確認できた．しかしその効果は 16 ビットまでという大きな集約を行ったにも関わらず，パケット到着の向上は 10％〜20％程度の効果しか得られないことが分かった．これは，アドレスの集約で説明したように，2 時間という大きな時間で集約を行えば，大きなフローの集約が行えるが，アセンブリタイムアウトという短い時間でのフローの集約が余り起こらないことが原因となっている．

3.5　考察

このトラヒック測定では，アセンブリタイムアウトを 10ms〜100ms という値に設定しないとほとんどアセンブリの効果を得ることが出来ないことが分かった．

これは工科大の内部 LAN では 100Mbps のインターフェースを持つ PC が利用されてい

るが，インターネットへの帯域が最大 6Mbps と狭くその帯域をすべての端末で共有しなければならないことが原因であると考えている．

数 ms の間隔でパケットが到着するフローもあったが，これらのトラヒック以上に大部分を占めるフローとして，ファイル共有のトラヒックがあった．これらのフローは，他のトラヒックを妨害しないように帯域制御されている．この制御方法は，インターネット接続ポイントにある帯域制御装置で TCP_ACK パケットのウインドウサイズを縮小することで行われている．ウインドウサイズの縮小は，パケットの転送の連続数を少なくすることに繋がり，パケットアセンブリによる負荷軽減が行えなくなる．

そこで，パケットアセンブリが有効に行えるネットワーク環境について考える．今後，トラヒックの増加とそれに対する WDM 技術等の発展によりリンク帯域は拡大される．工科大とインターネット境界の帯域は 6Mbps でありそれをすべてのフローが共有していたためそれぞれのパケット到着間隔が広がってしまう結果になったが，この帯域が例えば，100Mbps になればより，短い到着間隔でパケットが到着し，より短いアセンブリ待機時間でパケット連結が可能になる．

どの程度の帯域があればよいかについては，広帯域であるほどよいということになる．今後も帯域は広がっていくため，帯域拡大の技術が進むほどパケットアセンブリにとって効率的なネットワーク環境ができると考えられる．

マルチフローアセンブリのためのフローの集約について調査を行ったが，その効果はあまり得られないことが分かった．この対策としては，フローの集約の多くなる広帯域なネットワークのノードでマルチフローを行うことが有効であると考えられる．例として，企業間を結ぶネットワークやトラヒックの集中する IX でマルチフローを行うことが望ましい．

第 4 章

広帯域ネットワークシミュレーション

NOC のトラヒック調査から, パケットアセンブリは, より広帯域のネットワークで行なえばタイムアウトが短く効率の良い負荷軽減が行なえると考えられる. そこで, ネットワークシミュレータ OPNET Modeler[23] を利用して, 広帯域ネットワーク環境でパケットアセンブリを行なった場合のシミュレーション検証を行なった.

4.1 OPNET へのパケットアセンブリ機能の追加

OPNET は，PC 上で通信ネットワークやプロトコルのモデリングを行いそのシミュレーションや評価を行うことができるネットワークシミュレータである.

ここでは, パケットアセンブリのシミュレーションを行なうために,OPNET へ追加した機能について説明する. アセンブリ機能の追加はルータの QoS バッファ部分で行った．QoS バッファとは，ネットワーク QoS を実現するための 1 つの技術に使われるものであり，FIFO，PQ，WFQ，CQ といったキューイングを行うためにルータに備わっているものである [24].

4.1.1 パケット待機機能

エッジルータにおいてのコア網のインタフェースに追加する．宛先別にパケットをまとめるため WFQ（Waited Fair Quieing）のプログラムを使用した．通常ルータでは，前のパ

ケットの転送が完了していれば到着したパケットは効率化のためにキューイングせずにそのまま転送するようになっている．

一方，パケットアセンブリでは後続のパケットを待機させる必要があるため，パケットが転送可能状態であっても転送しないようにし，すべてのアセンブリを行うパケットをキューイングするようにした．また、アセンブリを行う先頭パケットが到着したときには，アセンブリ待機時間であるアセンブリタイムアウト値をセットする．セットしたタイムアウト値の時間が経過すればパケット連結後に転送処理を実行する．今回，タイムアウト値は，すべてのアセンブリにおいて共通値を用いることにしたが，パケットヘッダ等から読みとった値をもとに，フロー毎にアセンブリタイムアウト値を調節し，QoS 制御を行うことも考えられる．

4.1.2　パケット連結機能

転送開始時にキューから連続的にパケットの取り出しを行ない, パケットの連結を行う．連結は，先頭のパケットへ後続のデータのポインタを記録していく．連結する際に，連結した分のデータサイズ分だけ IP ヘッダの Total Length 部に増加分を加えたサイズを書きこむ．またこのパケットがアセンブリされたパケットであることが分かるように IP ヘッダにアセンブリフラグを立てる．

パケットの最大連結数は 5 個とした．これは，シミュレータの条件で 1 パケットの最大サイズが，8160Byte(65535bit) までとなっており，1500Byte × 5=7500Byte となりそれ以上のパケットを連結すると最大サイズを超えるためである．TCP_Ack パケット等のサイズの小さいパケットならば，より多くのパケット連結が可能であるが，今回は簡略化のため 5 連結までとした．

4.1.3 フロー毎のキューイング

他の宛先が異なるパケット同士がアセンブリされないように宛先アドレス別にパケットをキューイングする．マルチフローアセンブリでは，宛先アドレスの完全一致ではなく，先頭から途中まで（ネットワークアドレス部または，それより集約されたアドレス部）のアドレスが等しければ同一のキューへキューイングするようにする．WFQ がこれに近い機能を提供しているため，このキューイング方式をもとにアセンブリバッファの作成を行う．

4.1.4 アセンブリパケット分解機能（ディスアセンブリ）

エッジルータにおいてのアクセス網のディスアセンブリ機能をインタフェースに追加した．アセンブリされたパケットを元のパケットに戻す機能である．到着したパケットがアセンブリパケットである場合（アセンブリフラグが立っている）には，そのパケットから後続パケットのポインタをとりだし，キューイングする．また，TTL の減算は行うが，チェックサムはシミュレータでは定義されていないため行わない．

4.1.5 注意点

ディスアセンブリは出力キューで行われるためその出力インタフェースは MTU が大きくなっていないと MTU を超えるパケットを受け取ることになってしまう．

アクセスネットワークには、Ethernet を用いる．またコアネットワークには、Ethernet よりも MTU の大きいネットワークを想定する．Ethernet よりも MTU の大きいネットワークの例として、ATM の 9180Byte や Ethernet Jumbo Flame9000Byte などがある．

4.2　アセンブリによるコアルータ負荷軽減とパケット待機時間によるフローへの影響

パケットアセンブリにより，コアルータの負荷を軽減できることをシミュレーションにより示す．パケットアセンブリでは，一定期間エッジルータでパケットを待機させることによって，パケットに遅延を与えてしまう．そこで，実際にアセンブリを行いコアルータの負荷が軽減された場合についての以下のような検証を行なった．

4.2.1　シミュレーション目的

パケットアセンブリによるコアルータの負荷軽減とアセンブリタイムアウト値によるフローへの影響について調べる．

4.2.2　シミュレーションモデル

シミュレーションには以下の図ようなモデルを用いた．それぞれアクセス網には 100Mbps，コア網には，2.4Gbps のネットワークを用いた．RTT(往復伝播遅延) は約 30ms となっている．サーバとクライアントは，それぞれのエッジルータに接続され，エッジルータは 1 つのコアルータに接続されている．

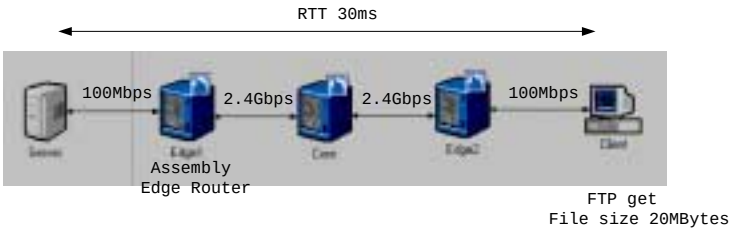

図 4.1　シミュレーションモデル

このネットワークで，クライアント側から FTP で 20MBytes のファイルの取得を行いサーバから転送されるデータパケットのアセンブリを行う．Edge1 の Assembly Edge Router で，アセンブリを行わない場合，およびアセンブリを行いタイムアウトを 1ms〜5ms

まで変化させた場合について測定を行った．

またクライアント，サーバそれぞれの受信バッファサイズは 8760Byte，パケット処理能力は 5000pps に設定した．これらの値はパケットアセンブリシミュレーションに影響するがその影響については 6.4 章に示す．

4.2.3　シミュレーション結果

シミュレーションの結果として，コアルータへ転送されるパケット数，コアルータの負荷，RTT，Throughput を図 4.2 に示す．

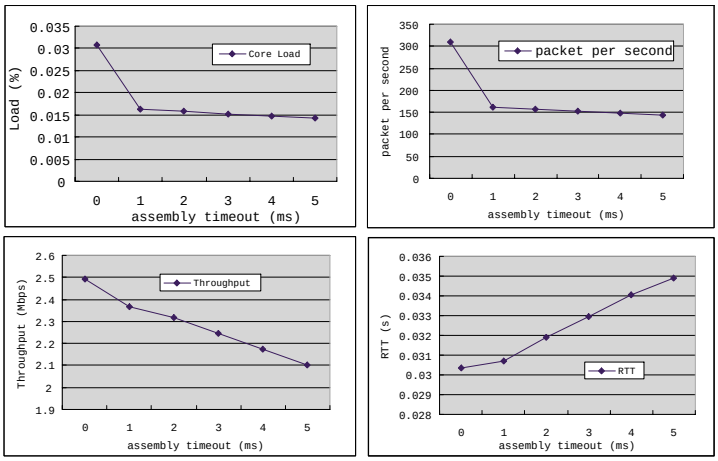

図 4.2　シミュレーション結果

まず始めに，コアルータのパケット数と負荷は，タイムアウト値を大きくするほど減少しており，パケットアセンブリによるルータの IP ヘッダ処理負荷軽減が確認できた．アセンブリタイムアウトを 1ms に設定した場合には，コアルータの負荷は約半分になっていることが確認できる．

しかし，2ms 以降は，コアルータの顕著な負荷軽減は見られなかった．次に RTT は，ほぼアセンブリタイムアウト値の分だけ増加していることが分かる．また RTT の増加に伴い, フローのスループットも低下している．アセンブリを行い 1ms の待機時間を与えたとき

のスループットの減少率は，アセンブリを行わないときと比較して約 5 %の減少となっている．

4.2.4　考察

パケットアセンブリによりコアルータの負荷軽減が行えることから，トラヒックが集中するバックボーンネットワークにこの方式を用いることで，ノード転送のボトルネックを解消でき，効率の良いネットワーク転送が行えること言える．

このシミュレーションでは，最大 5 個のパケット連結が行えるようになっていた．そのため通常転送と比べて 5 分の 1 の負荷軽減になるはずである．しかし，タイムアウトを大きくしてもその分の荷軽減が行えなかった．これは，TCP のパケット転送方法に関係している．TCP では，フローをコントロールする機能があり，UDP のようにパケットを転送し続けるわけではなく，相手端末からの応答を待ちながら転送を行う．

ここで，このシミュレーションでの端末から送信されるパケット連続数は最大は 6 個であった．この 6 パケットのエッジルータへの入力に対する出力パケットは，1 パケット (旧 5 パケット）＋1 パケット (旧 1 パケット) ＝ 2 パケットとしてコアに転送され，パケット数の変化としては，6packet → 2packet で 3 分の 1 になる．このことから，アセンブリを行う際に端末から一度に連続して転送されるパケット数が重要であることが言える．このシミュレーションのような場合では，連続する 6 パケットを 3 パケットずつにまとめることにして，タイムアウトを短く設定することも有効であると考えられる

また，TCP の場合，コアルータには，データとは逆方向に TCP_Ack パケットが転送されている．このシミュレーションでは Delayed_Ack を用いているため，データ packet の半分の Ack がコアルータに流れているが，このシミュレーションではそれらのパケットはアセンブリしていないため，コアルータの負荷は，全体として約半分の減少となっている．

ここで，コアルータの負荷が約半分になったということについて，これ以上またはこれ以下の負荷軽減が必要であるのか，もしくはこのままが良いのかはこの結果からは示すことができない．これは，パケットアセンブリを行う目的がバックボーンノードのボトルネック解

消であり，このシミュレーションではバックボーンがボトルネックになってないためである．どれだけの負荷を軽減すれば良いかについては 4.4 章で検証する．

次に，フローのスループットに着目すると，スループットが低下しまうことが分かった．これは，

$$Throughput \propto \frac{Windowsize}{RTT}$$

から，スループットが RTT に反比例するためアセンブリ待機時間が RTT に加算され，スループットの低下に繋がったと言える．しかし，それぞれのパケットがエッジルータに到着してから待たされる時間を考えると，すべてのパケットがアセンブリ待機時間分だけ待たされるわけではなく，後続のパケットほど待機時間は短くなるため，単純に RTT 値が大きくなることとは異なると考えられる．

このシミュレーションモデルでは１つのフローしか転送されていないためコアルータの負荷はアセンブリを行わなくても 0.03%と非常に低くノードのボトルネックは起こっていない．このためヘッダ処理削減による効率化の影響はほとんど見られず，アセンブリタイムアウトによるスループット低下の影響のみが結果として現れた．

パケットアセンブリの有効性としては，トラヒックが増加し，ルータの負荷が 100%を超え，パケットロスが起こることによって，パケット再送による無駄なトラヒックが発生する．そこで，パケットアセンブリを行い，フローの数%のスループットが減少しても，パケットロスの少ないネットワーク転送が行えれば，パケット再送の無駄なトラヒックが減少する分，通常のネットワーク転送より効率的な転送が行えると考えられる．

4.3 TCPAck パケットのアセンブリ

上のシミュレーションでは，データ方向とは逆方向に転送される TCP_Ack パケットのアセンブリは行わなかった．Ack パケットをアセンブリすることは、そのパケットの遅延からパケットロスにつながる．その理由として，TCP_Ack パケットは，データが到着したことを相手端末に知らせる役目がある．この Ack パケットがアセンブリによる遅延により，TCP 再送制御のタイムアウトとなり，データが届いていないと判断された場合，データの

再送が起こり，転送の無駄が生じてしまう．

しかし，トラヒック調査の結果から Ack パケットは小サイズのパケットであり，これらをまとめて転送することは，バックボーンルータの大きな負荷軽減になる．

4.3.1　目的

Ack パケットのアセンブリによる遅延が，パケットロスとして扱われてしまわないか，また Ack パケットをアセンブリすることによる，負荷軽減の効果について調査する．

4.3.2　シミュレーションモデル

シミュレーションモデルは，図 4.3 のように上のモデルと同様のモデルを用いた．

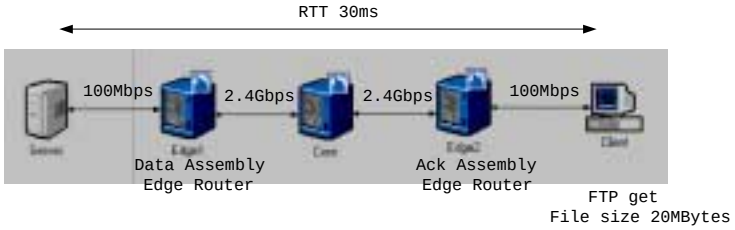

図 4.3　シミュレーションモデル

このモデルでは，FTP のデータ転送方向のアセンブリに加え，反対のエッジルータ (Edge2) にもアセンブリ機能を実装し，Ack のアセンブリが行えるようにする．また Ack パケットのアセンブリの条件は，データパケットのアセンブリ条件と同様である．

4.3.3　シミュレーション結果

シミュレーションの結果として，コアルータへ転送されるパケット数，コアルータの負荷，RTT，Throughput のグラフを図 4.4 に示す．

Ack をアセンブリすることでより Core ノードの負荷を軽減された結果が得られた．また Ack パケットをアセンブリしたことにより，遅延のために再送制御が起こっていないかを調

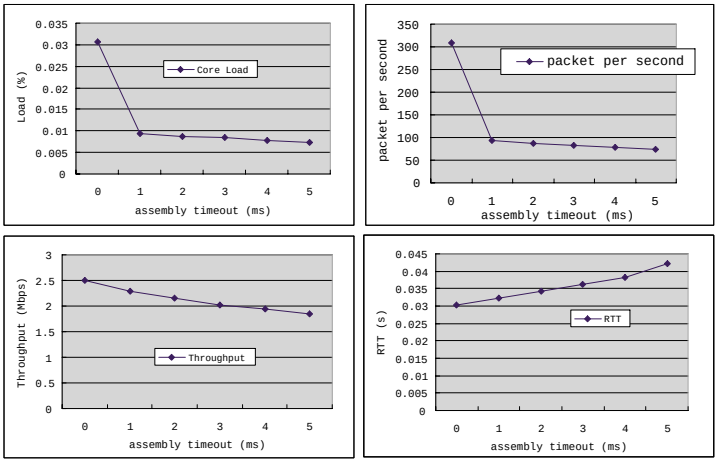

図 4.4 シミュレーション結果

べたところ，再送制御はおこらず，Ack パケットの遅延による再送制御の悪影響は見られなかった．

しかし，Data パケットと Ack パケットの両方で待機させた分の遅延による RTT の増加とスループットの低下が表れている．

4.3.4 考察

Ack パケットの遅延による再送制御が起こらなかった理由としては，アセンブリのタイムアウトの 1ms〜5ms に対して，TCP の Ack パケットの再送タイムアウトの時間が初期値で 1 秒，最小値が 0.5 秒と十分大きかったたためである．また，この再送タイムアウト値は一般的な TCP の設定である．よって，アセンブリにより Ack パケットにアセンブリ待機時間程度の遅延が与えられることはほとんど問題がなく，Ack をアセンブリすることによりさらに大きな負荷軽減ができることから，Ack パケットのアセンブリは有効であるといえる．

今回のシミュレーションでは，伝送エラーによるパケットロスはシミュレートしていなかった．TCP_Ack は，どこまでのデータを受信したかを知らせるようになっているため，単体でロスした場合は，再送タイムアウト以内に後続の Ack パケットが到着すれば再送が起こることはない．

しかし，過度の Ack のパケットアセンブリを行った場合に，アセンブリされたパケットがロスした場合の影響については今後検討する必要がある．また，ロスが起こっても次の Ack がカバーできる程度の適度な Ack パケットアセンブリについて考える必要がある．

4.4　コアルータがボトルネックになっている場合のシミュレーション

4.4.1　目的

ここまでのシミュレーションでは，コアルータに 1 つのフローの負荷のみがかかっていた．CPU の負荷が低い状態では，アセンブリを行い負荷軽減が行えても，結果としてスループットが低下するだけになる．そこでノード負荷が高い場合のアセンブリによる効果を示すため以下のようなシミュレーションを行った．

4.4.2　シミュレーションモデル

実際に他のトラヒックが流れていることを想定して，コアルータをボトルネックにしたシミュレーションを行う。

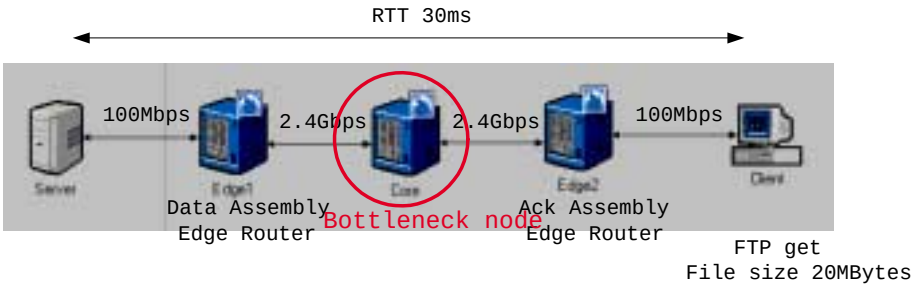

図 4.5　ノードボトルネックシミュレーションモデル

4.2 章，4.3 章と同様のシミュレーションモデルを用いた．ボトルネックノードとしてコアルータのパケット処理能力は 2500pps に設定している．また最大連結数が 5 パケットであり，アセンブリタイムアウト時間を 1ms〜3ms までにしている．またコアルータへの負荷を

かけるために FTP のスループットが高くなるようにウィンドウサイズを 8160 から 32768 へと変更している．その他のパラメータはこれまでのシミュレーションと同様である．

4.4.3　シミュレーション結果

シミュレーションの結果として，コアルータの負荷と Throughput のグラフを図 4.6 に示す

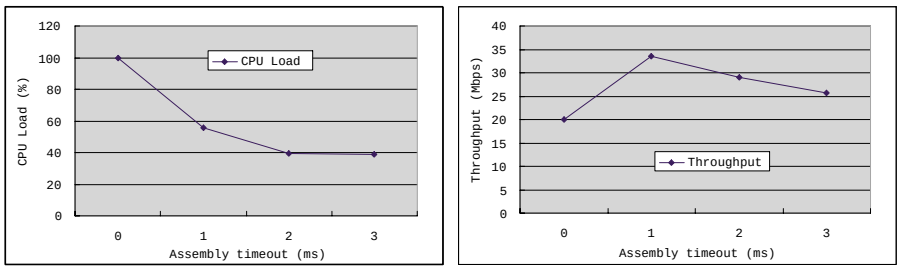

図 4.6　シミュレーション結果

パケットアセンブリを行わない場合，コアルータの性能 (packet per second) が低いため，その負荷は 100%になっている．アセンブリを行った場合，CPU の負荷は 55.1%まで減少した．さらにタイムアウトを大きくすると，減少は見られるが，これまでのシミュレーションと同様に顕著な負荷軽減は見られなくなる．

次にスループットについては，アセンブリを行わないときには約 20Mbps である．アセンブリを行ったときには，1ms の時で通常よりも高い 33.44Mbps のスループットが出ている．2ms 以降はアセンブリを行わないときよりは高いスループットが出ているが，タイムアウトが大きくなるに従い減少していく．

4.4.4　考察

TCP のみのフローが転送されていたため，ルータの負荷が 100%になってもフロー制御により，それ以上のトラヒックは転送されず，バッファ溢れによるパケットロスは無かったが，UDP のようなフロー制御の行えないフローが混在するときには，バッファ溢れが起こ

り，それにより TCP のパケット再送制御等が起こり効率の悪いネットワーク転送になると考えられる．

次に，アセンブリを行った場合のスループットの向上は，コアルータの CPU 処理のボトルネックが解消され CPU 負荷が 55.1%となったことによる結果である．

しかし，それ以降アセンブリタイムアウトを大きくすることでスループットが低下することについては，コアルータの負荷がすでに 100%以下になり，それ以下に負荷が下がらなくても十分パケットを処理できるためである．これは，このモデルでのエッジルータの負荷軽減（パケット待機時間を 2ms 以上に設定すること）を行いすぎであると考えることができる．負荷軽減のためのフローへの影響があるため，コアルータの負荷を軽減しすぎることは避けなければならないため，アセンブリを行う際にそのバランスについて調整する必要がある．

またネットワーク毎にトラヒックの特性が異なり，さらに時間帯によっても特性は異なるため、それぞれのネットワークに適したそれぞれの場合に適したアセンブリの設定が必要であると考えられる．

例えば同じネットワークでも，ネットワークがあまり使用されておらず負荷の低い場合には，アセンブリタイムアウトを短く設定しすることでそれぞれのスループットを高くし，ネットワークの利用が多く負荷が高いときには，アセンブリタイムアウトを長く設定することで，コアルータのボトルネックを解消し高いスループットを得る方法が考えられる．

4.5　マルチフローマルチフローアセンブリの評価

アセンブリの悪影響を小さくするために，タイムアウトを小さくすことを考える．ここでは，複数の行き先が同一であるフローをまとめることによって効率的にアセンブリを行うマルチフローアセンブリのシミュレーションによる検証を行う．

4.5.1　目的

アセンブリによるスループットの低下が問題となり，より短いタイムアウトの設定で，アセンブリを行なうことが望ましいことが分かった．

しかし，短いタイムアウトでは，多くのパケットをまとめることができない．そこで，提案している複数の宛先が同一であるフローを 1 つのフローと見て，アセンブリを行なうマルチフローアセンブリの有効性を示す．

4.5.2　シミュレーションモデル

マルチフローアセンブリの有効性を示すため, 図 4.7 に示すシンプルなモデルを用いる.2 台の FTP クライアントでそれぞれ 20Mbytes のファイル取得を行う 2 つのフローを 1 つのフローとして扱いマルチフローアセンブリを行う．またデータと Ack の両方のパケットアセンブリを行う．

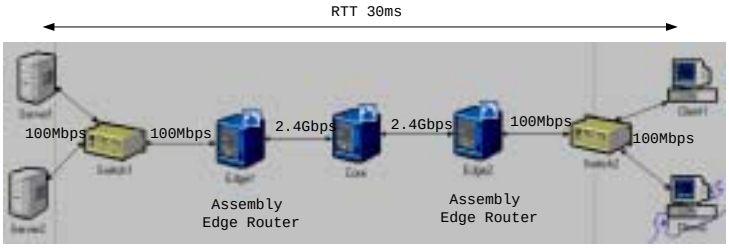

図 4.7　シミュレーションモデル

4.5.3　シミュレーション結果

シングルフローアセンブリとマルチフローアセンブリの比較で，アセンブリ待機時間を短く変化させた時のコアルータの負荷とスループットの変化を図 4.8 に示す．

シングルフローの場合は，タイムアウトを短くしていくと，エッジルータのアセンブリバッファに蓄積されるパケット数が少なくなり，パケット連結が行えなくなるため，コア

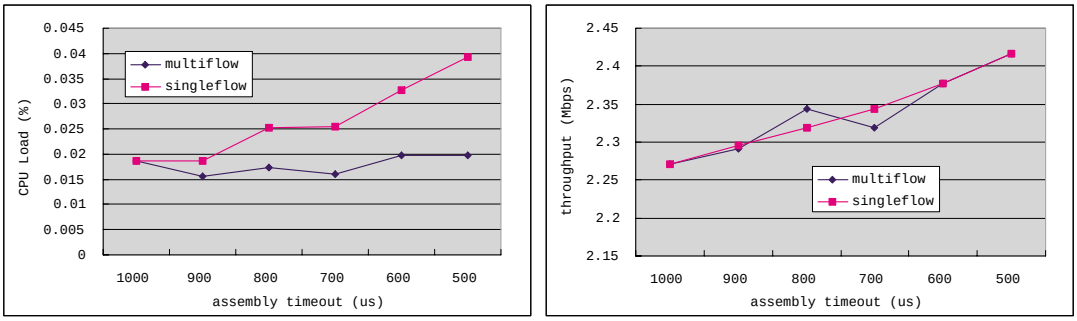

図 4.8　シングルフローとマルチフローの比較

ルータの負荷は上昇していく．

しかし，マルチフローの場合は，両方のフローのパケットを同一のバッファに蓄積するため，タイムアウトが短くなっていってもパケットの蓄積数の不足（最大連結数５パケット）はないため，コアルータの負荷軽減が行えることが分かる．またスループットのグラフを見ると，シングルフロー，マルチフローの両方とも，タイムアウトを短くすることで，スループットが上昇してくことが分かる．

4.5.4　考察

マルチフローアセンブリにより，タイムアウトが短い場合にも負荷軽減が行え，効率的にアセンブリが行えることが分かった．待機時間を 500 μ s に設定したときでも，1ms とほぼ同等の負荷軽減ができ，スループットについては 2.42Mbps となり，アセンブリを行わないときのスループット (4.2.3 章) の 2.49Mbps に近づく結果になり，フローへの影響を軽減できることが分かった．

また，シングルフロー，マルチフロー両方の方式でスループットがほとんど変わらないことについては，パケット到着率が上がってもアセンブリタイムアウト値の時間だけは必ず待機させられるためである．これには，アセンブリパケットの転送トリガーにパケット連結数による制限を加えることにより，よりスループットの高くなるマルチフローアセンブリが行

えるようになると考えられる．

今回のシミュレーションでは，２つのフローを同時に転送したが，それ以上のフローをマルチフローアセンブリすることができれば，よりタイムアウトを短く設定した効率の良いアセンブリが行え，フローへの影響の少ない負荷軽減が可能であると考えられる．

しかし，各端末の通信相手は様々であるため，同時刻にかつ同じネットワークへパケットが転送され，マルチフローが行えるかを検証する必要がある．また，フローの集約方法として，多くのフローが集まる IX にマルチフローアセンブリを用いることや，IPv6 のアドレス体系を用いたアドレス集約，さらにはエッジルータ間の通信によるフロー集約などを検討し，効率的なアセンブリを行うことが今後の課題となる．

第 5 章

パケットアセンブリシステム (PASTA)

シミュレーションと並行して，アセンブリシステム装置の開発を行った．

5.1 概要

PASTA（Packet Assembly System by TAO）はルータ機能に加えて，パケットアセンブリを行うためのバッファを装備し，フロー毎にパケットを待機させ，アセンブリタイムアウトや MTU（最大転送単位）の制限をトリガーにパケット連結を行い転送する装置である．

また，本装置では，複数のパケットアセンブリの形式を実装している．提案されているア～エの４つのパケットアセンブリの形式を図 5.1 に示す．ア，イのカットアセンブリは，ま

図 5.1 アセンブリの形式

とめるパケットの IP（ア）または TCP/IP（イ）ヘッダの一部の情報（バージョン等の冗長な情報）をカットすることで，転送量を削減する形式である．ウは，それらのカットを行

わずにまとめる形式である．エはカットを行わず，それぞれのパケットの間にマーカを加える形式である．アセンブリ・リアセンブリ（パケットの分離）の高速化やアセンブリ後のパケットの転送量等の比較を行うことにより，最適な形式について検討を行っていく．

本装置を用いて，DV(Digital Video) トラヒック転送と負荷トラヒック転送を行う実験では，パケットアセンブリを行うことで，バックボーンルータの負荷軽減が行え DV 品質の向上が行えることを示している [8].

また，PASTA によるコアルータ負荷軽減の検討に加え，PASTA は複数のパケットの連結方法を実装しいるため，それぞれの方法についての検討が行われいる [19].

これらの実験検証で，PASTA では，MPU の処理能力不足とそれを補うバッファのサイズ不足等の問題から，高スループット時には，アセンブリを行わないときでもパケットロスが発生することがわかっている．

現在，この処理能力の向上のために，PASTA のボトルネックとなる部分の発見を行い，その改善を行うために PASTA2 の開発を行っている．これにより，より負荷の高いトラヒックを転送するときの実験が行えるようになると考えている．以下では，PASTA を用いたファイル転送 (FTP) の負荷軽減の実験につき報告する..

5.1.1　ネットワークモデル

図 5.2 のようなネットワークモデルを用いた．アクセスネットワークに Ethernet，コアネットワークに ATM を用いた．PC-エッジルータ間は，100Mbps Ehternet で接続し，ATMSW-PASTA 間は OC-3(155Mbps) の ATM で接続されている．シミュレーションと同様に FTP クライアントでファイルの取得を行う．アセンブリについては，FTP サーバからのデータを PASTA1 でアセンブリし，PASTA3 のコアルータを経由して PASTA2 でディスアセンブリされ FTP クライアントに転送される．ATM スイッチは上記のような転送経路になるようにパスの設定を行っている．

パケットアセンブリでは 2 つのタイムアウト方式が考えられていることを述べたが，先頭パケットが到着してからタイムアウトまで後続のパケットを待ちつづけるシミュレーション

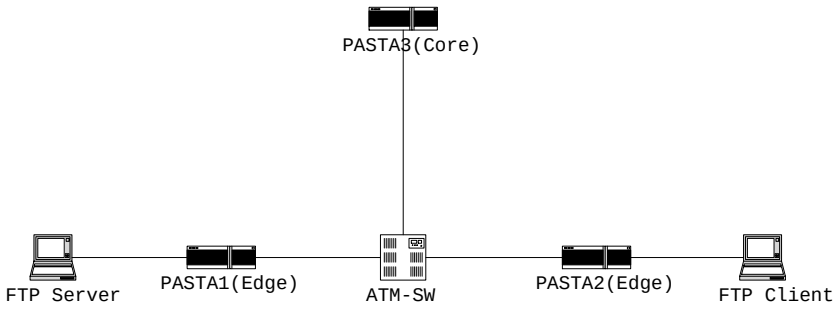

図 5.2 ネットワークモデル

で用いた方式とは異なり，PASTA の転送のトリガーには，パケット到着間隔がタイムアウト以上であれば転送するという条件を用いている．PASTA2 には両方のタイムアウト方式を実装することにしている．

5.1.2 結果

スループットとコアルータの CPU 負荷を図 5.3 に示す．シミュレーションでの結果とは異なり，横軸がタイムアウトではなく最大連結数を示している．これはアセンブリを行わずにそのまま転送する 1 パケットの場合から最大 5 パケットまで連結したときの結果である．アセンブリ最大連結数を大きくするほど，コアルータの負荷軽減が行えていることが分か

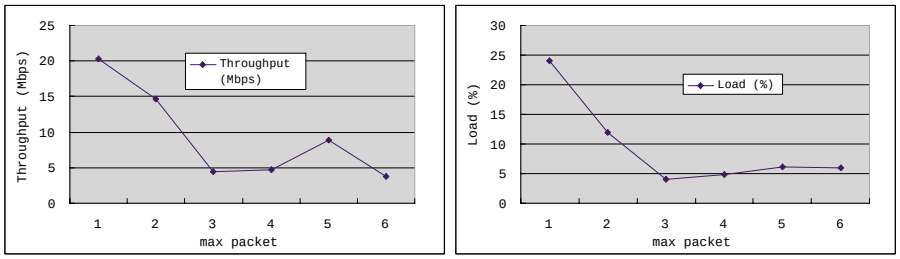

図 5.3 実機測定結果

る．しかし，それに伴うスループットの大きな低下が見られた．最大連結数を 3 パケットにした場合には，スループットは，4 分の 1 程度まで低下し，アセンブリによるフローへの大

きな影響が出ていることが分かる．

5.1.3　考察

コアルータの大きな負荷軽減が見られたが，これはパケット数の減少によるものだけではなく，アセンブリ待機時間により，ファイル転送のスループットが低下したことによる結果も含まれている．

この原因としては，RTT を計測すると数 ms であったことがあげられる．RTT に対するアセンブリ待機時間の割合が大きくこの場合では，アセンブリを行うことでアセンブリ待機時間の元の RTT に対する割合が大きく，パケットアセンブリによる合計の RTT が大きくなりスループットが低下したと考えられる．

今回の実験では，アセンブリを行わないときの RTT を大きくすることが出来なかった．これに対しては，このモデルのバックボーン部分に遅延を発生させる装置を挟むことや，実際のネットワークのエッジルータに設置し検証する必要がある．

第 6 章

まとめ

この章では，本研究の目的であるパケットアセンブリによる負荷軽減とそれによるフローへの影響について得られた成果を整理し，今後の課題について述べる．

6.1 パケットアセンブリによる負荷軽減

インターネットの急激なトラヒック増加によるノードボトルネックを解消するためのパケットアセンブリ方式の提案を行った．

実際に高知工科大学ネットワークのトラヒック調査を行い，インターネットへ小サイズのパケットが転送されていることを示した．ネットワークシミュレータに提案しているパケットアセンブリを実装し，シミュレーションを行うことで IP ヘッダ処理の負荷軽減が行えることを確認し，トラヒックの集中するバックボーンネットワークに本方式を用いることが有効であることを示した．

また，実際にボトルネックになっているノードにパケットアセンブリを用いたシミュレーションでは，ボトルネックを解消し，通常転送よりも高いスループットでデータ転送が出来ることが分かった．さらに，この実験から，パケットアセンブリによって多くのパケットをまとめるほど効率が良いネットワーク転送が行えるのではなく，アセンブリ待機時間とコアルータ負荷のバランスが重要であることが分かった．

光スイッチングへの移行

6.2　パケットアセンブリによるフローへの影響

パケットアセンブリによるパケット待機時間が TCP のスループットに影響することを示した．特に PASTA の実験では，RTT が小さくそれに対するアセンブリタイムアウト値が大きかったため，アセンブリによる大きな影響が出た．このため RTT の短い LAN 内のバックボーン網より，RTT の大きいインターネットを介した ISP や IX でのパケットアセンブリが有効であると考えられる．

しかし，一般的に通信を行う上で，RTT は小さい方が望ましい．そこで，アセンブリによるフローへの影響を軽減するために，マルチフローアセンブリを行うことによりフローへの影響が少なく，より効率の良いアセンブリが行えることを示した．

6.3　パケットアセンブリの有効性

以下では，これまでの実験結果等からパケットアセンブリの有効性について考察する．

6.3.1　パケットアセンブリによる負荷軽減の目標

パケットアセンブリの目的は，近年のトラヒック増加がプロセッサ処理能力のムーア則に頼ることができないため，コアルータのプロセッサ負荷軽減であった．そして，今回のパケットアセンブリシミュレーションによるコアルータの転送負荷は通常の転送の３分の１に軽減できることが分かった．

ここで，ムーアの法則による CPU 処理能力の向上とトラヒック量の増加を予測するギルダの法則（1 年で 3 倍），さらに負荷軽減後のトラヒック量（パケット数で考えた場合）のグラフを図 6.1 に示す．CPU 処理能力がトラヒックの増加をカバーできないことが分かり，ノードがボトルネックとなっているネットワークに本方式を用いることでその時点でのボトルネックを解消は行えると考えられる．しかし，その数年後には急激なトラヒックの増加により再びボトルネックが起こると考えられ，より大きな負荷軽減が必要になる．

今回のシミュレーションでは 3 分の１までの負荷軽減しか行っていなかったが，たとえよ

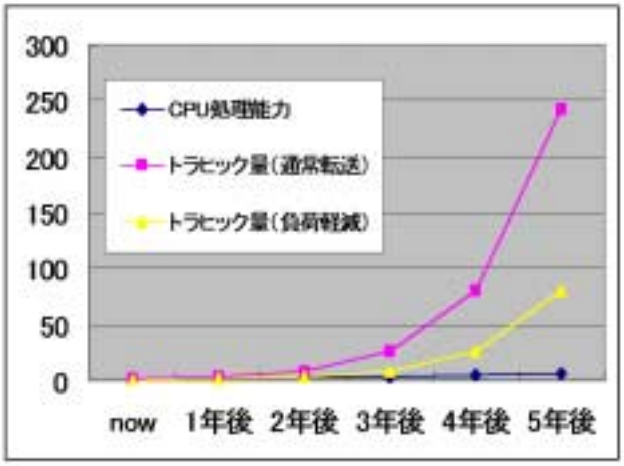

図 6.1 CPU 処理能力とトラヒック量の増加

り小サイズのパケットをまとめることができ，MTU のパケットサイズまでまとめることでより大きな負荷軽減が行えても，この急激なトラヒック増加に伴う負荷軽減を行って行くことは難しいと考えられる．また，このグラフでは現在の処理性能とトラヒック量の両方を１とし，ボトルネックとなっていない状態を想定しているが，現在のネットワークでは，コアルータが過負荷にならないようにアクセス網で制御されたトラヒックが転送されていることを考えるとすでにコアネットワークがボトルネックになっていると言える．

インターネット人口の増加，常時接続，高画質映像ストリーミング，ファイル共有など，帯域が広くなればなるほどトラヒック増加につながる要因が多く存在する．どれだけの負荷軽減を行えば良いかについては，利用者にどれだけのトラヒック量を転送させたいかを考え，それらのトラヒック集中によりコアルータがボトルネックにならないようにパケットアセンブリによる負荷軽減を行うべきであると考えられる．

6.3.2 プロセッサの有効利用

トラヒックの増加に対応しより多くのデータ転送を可能にすることとは異なるもう一つのパケットアセンブリの利用方法について考える．パケットアセンブリは，最大連結数を大きく，パケット待機時間を長く設定することにより，より多くのパケットをまとめ，大きなコ

アルータの負荷軽減が可能になるが，パケットアセンブリをコアルータの負荷が混雑時に 100%にならないようにするための機能として使用することも可能である考えている．トラヒックの変動幅の大きい様子を表したグラフを図 6.2 に示す．

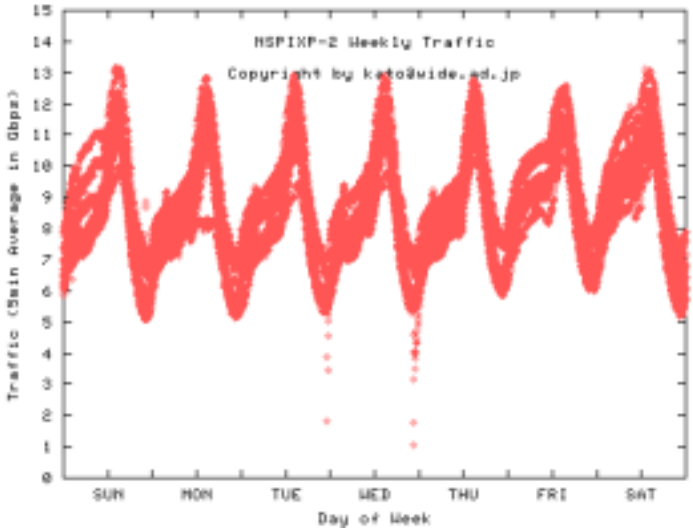

図 6.2　1 週間のトラヒック変動

トラヒック量が少ないときで 5Gbps 程度，多いときで 13Gbps のトラヒックが転送されており大きな変動があることがわかる．このグラフでは，1 日の周期での変動がわかるが，より短い周期（時間，分，秒）で見た場合でも大きな変動があることもわかっている（自己相性）．

ネットワークの設計では，トラヒックの混雑時を想定して余分を持たせると，混雑しないときのネットワーク利用率が悪く無駄な資源になってしまう．通常はパケットアセンブリを用いずにそのまま転送を行い，混雑時にパケットアセンブリを用いることで，有効なネットワーク資源の利用ができると考えられる．

また，QoS を保証するための帯域制御等は，ネットワークの資源（ここでは，ルータのプロセッサ負荷）を 100%利用している状態では行いにくいが，パケットアセンブリによりネットワーク資源に余裕を持たせることが出来ることから，帯域制御が実行しやすくなると考えられる．（プロビジョニング [25]）

6.3.3 フォトニックルータへの応用

次にプロセッサの負荷軽減とはことなる利用方法として，電気的処理を行わずに光りのままで転送を行うことにより高速化を図るフォトニックルータへの応用について考察する．現在，電気的なルーティング処理の限界から，転送されてくるデータを光から電気への変換を行わず光のままで転送を行うフォトニックルータの研究が行われている．フォトニックルータでは，受信するパケットの電気変換を行わず光のままで方路の切り替えを行う必要がある．その切り替えには，入ってくる光をミラーを動かし反射させることや，バブルを発生させることで光の方路を切り替えを行う方式が考えられている．現在の光スイッチング処理は，数十 ms 程度であり，切り替え速度が低速なため，このスイッチング技術利用方法としてはバックアップ回線への切り替え等に使用されるが，フォトニックルータを実現するためには，パケットレベルでのスイッチングが行えるようにスイッチング速度の向上が必要である．よって，スイッチング速度が遅いままであると，方路の切り替えの期間は，データを転送することができないため，短いパケットが多く転送される場合は切り替えが頻繁に起こり帯域の無駄が生じてしまう．

ここで，パケットアセンブリを用いることによってパケットサイズを大きくすることで，切り替えの頻度を少なくすることにより，効率的な帯域利用が行えるのではないかと考えられる．

6.4 パケットアセンブリの効率に関わるパラメータ

トラヒック調査やシミュレーションを行う上で重要であると考えられたパケットアセンブリの効率に関係する TCP ウインドウサイズ，トラヒックの混雑，アセンブリバッファ数，端末の処理速度，MTU について以下に述べる．

6.4.1　TCP WindowSize

TCP で通信を行なう場合，一度に WindowSize に示されているデータ量を超えて送信することはできない．Ack パケットが返ってくることによりそれ以降のパケットの転送が可能になる．これは，TCP が帯域に適応してスループットを調節するための機能である．よってエッジルータでより多くのパケットをまとめるためにアセンブリタイムアウトを大きく設定しても，WindowSize 以上のパケットをアセンブリすることはできないことになる．

通常のフローの WindowSize では，アセンブリを行なうコアネットワークの MTU より十分大きな値であるため，WindowSize がアセンブリ連結の制限になることはないと考えられる．しかし，WindowSize は，端末間の TCP の制御だけでなく，帯域制御装置などにより最大サイズが制限される場合がある．その値が小さい値に設定されている場合には，WindowSize がアセンブリ連結の制限になることが考えられる．

また TCP は通常，スロースタートという方式を用いており，常に WindowSize 分のトラヒックを転送しているわけではないためこの場合は windowsize ではないが一度に端末から転送されるパケット数が制限になること考えられる．

6.4.2　RTT

シミュレーションや PASTA での検証からも分かるとおり，RTT の大きいネットワークほどアセンブリの待機時間による影響が小さく，RTT が小さいほど待機時間の影響は大きくなる．

6.4.3　他のトラヒックとの混雑

1 つの回線を 1 つのフローのみが利用していた場合，アセンブリを行なうエッジルータに到着するまでに，他のパケットが同じ回線に割り込むことによって，割り込まれたフローのパケットの到着間隔を広げることが考えられる．今回のシミュレーションでは，図 6.3 のように単一のフローでの実験を行った．

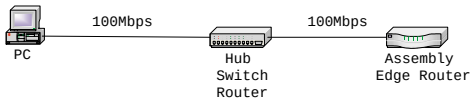

図 6.3　回線を単一フローが流れている場合

しかし，通常のネットワークのエッジルータでは，図 6.4 のように，複数の端末のフローがまとめられたトラヒックが到着するため，トラヒックの混雑が起こりパケットの到着間隔は大きくなる．

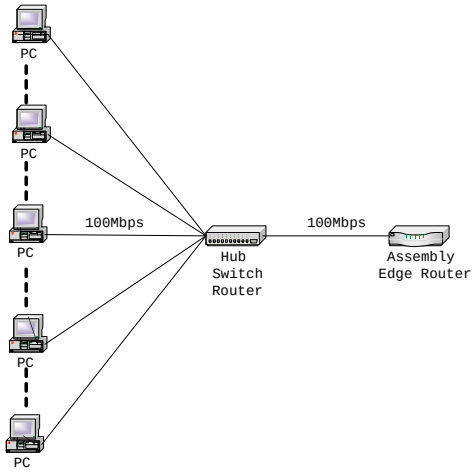

図 6.4　複数フローが流れている場合

回線が 100Mbps の場合，1 パケット（Ethernet の MTU1500Byte）の転送に必要な時間は，

$$\frac{1500Byte \times 8bit}{100Mbps} = 0.00012 = 120\ \mu\ s$$

より 120 μ s であるため，例えば，3 つのフローのパケットを順番に転送することを考えると，通常は 120 μ s の間隔でパケットが到着していたものが，残りの 2 つのフローの転送が混ざることによって，360 μ s 毎の到着になり、フロー別パケットの到着間隔が広がってしまうことが分かる．

従って，アセンブリエッジルータを設置する場所を，ISP や IX のエッジルータとした場合に，多くのフローが集約された回線であるため，この影響は大きいと考えられる．

6.4.4　アセンブリバッファ数

フローの数だけキューが必要となるものにキューイング方式である WFQ がある．WFQ では，フロー数が多くなるにつれて，管理するキューの数が増え，その負荷により性能が低下することが知られている [9].

パケットアセンブリも WFQ と同様にフロー毎にパケットを管理する方式である．特にアセンブリを用いる ISP や IX のエッジルータのフロー数は多くなると考えられる．よって，フロー数と負荷の関係について検証し，またフローの集約により，アセンブリのためのキュー数を減らす必要がある．

これらの負荷に対する実験は，シミュレーションでは行えないため，開発を行っている PASTA を用いて，フロー数を増やした場合のキュー管理負荷について検証する必要がある．

6.4.5　端末の処理速度とパケット転送間隔

今回のシミュレーションでは，PC のパケット転送性能（packet per second）は 5000pps に設定していた．これは，Ethernet の MTU の 1500Byte のパケットサイズで転送を行った場合を考えると最大で，12000bit(1500Byte) × 5000pps = 60Mbps までしか転送できないことになる．

現在では，端末が 100Mbps,1000Mbps のインタフェースを実装していても, アプリケーションの負荷や PC の性能の問題によってパケットを生成でず，それらの速度が出ていない端末は多い. 端末からのパケットの転送速度は, 帯域幅だけでなく PC の CPU の性能や, ネットワークインタフェースの性能にも影響される. 端末からのパケットの転送速度が高速になれば，エッジルータへのパケット到着間隔も狭くなり，より短いアセンブリ待機時間で多くのパケットをまとめることが可能になる．

6.4.6 MTU

現在，存在するネットワークには ATM，Ethernet ジャンボフレーム，FDDI などが存在する．MTU 以上のパケットの連結は行えないため，この値はパケットアセンブリの最大連結数の制限となる．今後，転送速度の向上や，ロス率の低減により，MTU の大きいネットワークが登場すれば，多くのパケット連結が行えることになりそれだけの負荷軽減が可能となる．例として，IP over WDM が可能になれば，IP の最大 MTU での転送が可能になる．

6.5 今後の課題

今後の課題としては，以下のものがあげられる．

6.5.1 ISP，IX 等のネットワークトラヒックをモデルとしたパケットアセンブリ

今回のシミュレーションでは，WFQ で宛先アドレスまた宛先ネットワークアドレス別にパケットをキューイングすることが出来ず，多端末を用いたシミュレーションを行うことが出来なかったため，負荷が高くなると考えられる ISP や IX 等のネットワークモデルを作成しそれらのシミュレーションを行うこと出来なかった．

今後この機能を動作させることにより上記のシミュレーションを行うことでアセンブリの有効性を示す必要があると考えている．また，これらのモデルで，マルチフローアセンブリのシミュレーションを行うために，アドレス別のキューイング機能を応用して，ネットワークアドレス別のキューイング機能を実装する必要がある．これらのシミュレーションを行う上で問題となることの一つに，アセンブリバッファ管理がある．

パケットアセンブリでは，フロー毎にバッファを管理するため，ISP や IX 等のトラヒックの集中するネットワークでは多くのフローを管理しなければならない．WFQ では，フローを管理する数が多くなると効率が悪くなるという問題もあため，実際に多くのフローがアセンブリエッジルータに到着したときにパケットアセンブリがどのような影響を与えるか

について検証する必要がある．

6.6　RTT とアセンブリタイムアウトの関係

パケットアセンブリでは，RTT が大きければ影響は小さく，RTT が小さければアセンブリの影響が大きくなる．RTT を 10ms,20ms,30ms,40ms,50ms と変化させることによりそれぞれのネットワークでどのようなパケットアセンブリタイムアウトによるフローへの影響が見られるかを調査する必要がある．

6.6.1　パケットアセンブリへの QoS 制御機能の追加

パケットアセンブリでは，パケットの到着率の高いフローほど効率的な転送が行える．工科大トラヒック調査からも分かるように，インターネットにおいては様々な速度のフローが転送されている．転送速度の速いフローほど効率の良いパケット転送が行われ，転送速度の遅いフローほど効率の悪いパケット転送が行われることになる．

これについては，最大パケット連結数をトリガーにする転送は行わずに，アセンブリタイムアウトによってのみ転送を行うことで不公平性をなくすことが考えられる．このようなアセンブリ後のパケット転送についての QoS 処理については今後検討する必要がある．

6.6.2　マルチフローアセンブリのためのフローの集約について

トラヒック調査では，測定を行ったネットワークの帯域の問題に加え，フローの集約がうまく行えないことが問題となった．フローの集約は，アセンブリ待機時間を短し効率のよいアセンブリを行うことや，キュー数の増加による管理負荷の軽減するためにも重要である．ここでは，より効率的なフロー集約について考える．

1 つの方法として，IPv6 のアドレス階層構造のフロー集約によるパケットアセンブリの効率化があげられる．IPv6 では，経路エントリー（登録されたアドレス情報）が集約できるように図 6.5 のように階層化アドレスになっている．

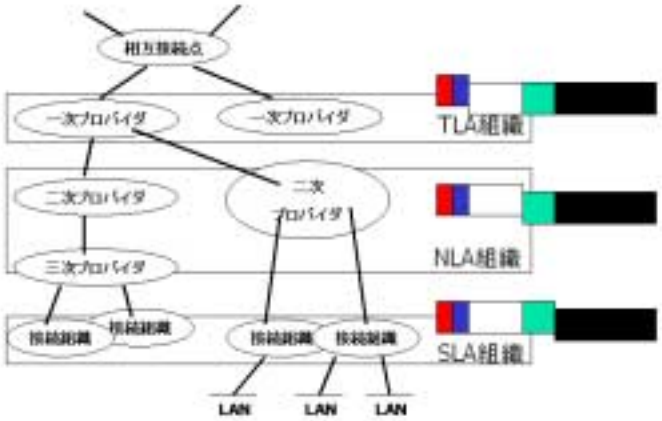

図 6.5 IPv6 の階層別アドレス割り振り

TLA（Top-Level Aggregation Identifier，最上位階層集約識別子）までの上位 16 ビットで最上位 ISP を識別，24 ビットの NLA（Next-Level Aggregation Identifier，次階層集約識別子）は複数のレベルの下位 ISP を識別，そして下位 48 ビットがユーザに割り当てられ，各ユーザは 16 ビットは SLA（Site-Level Aggregation Identifier，サイト階層集約識別子）を用いてサブネットを設計できるようにアドレス構造が定義されている．このように階層化を行うことによって，TLA を取得した組織の外からは，それよい下位の SLA や NLA の情報は見えずに，大きな 1 つの TLA が存在するものとして見えることになる．これにより，TLA を取得した組織外のルータの経路表では，TLA を取得した組織が 1 つ存在するように見えるだけで，そのネットワーク中にたくさん存在する SLA や NL の存在は見えないようになおり，ルータ経路表のエントリが少なくなる．そこで，相互接続点（IX）でアセンブリを行う場合には，アドレスの TLA の部分のみを参照し，それらのフローをマルチフローアセンブリにより集約すれば，1 次プロバイダの数だけのフロー数で処理することが可能である．ここで問題となるのは，様々なアプリケーションがある中で，それらをすべて同じ扱いをして良いかという QoS の問題がありそれらを検討する必要がある．

謝辞

本研究を進めるにあたり，指導教員として終始御指導，御助言していただいた島村 和典教授に深く感謝致します．また，在学中親切なる御助言を頂いた清水 明宏 教授，浜村 昌則 講師をはじめとする諸先生方に心より感謝致します．そして，島村研究室サブグループ（QoSグループ）のリーダーとして直接研究指導にあたり，有益な議論をしていただいた，通信・放送機構 高知通信トラヒックリサーチセンターギガビットネットワーク研究開発プロジェクトの神田敏克フェロー（元研究員）をはじめとするメンバーの皆様にも感謝致します．最後に，同じ研究室で研究を進めてきた谷岡亮介君，山田敦君をはじめとする島村研究室の皆様にも感謝致します．

参考文献

[1] 神田，島村；パケットサイズ制御によるルータでのデータ転送効率向上に関する考察, 信学技法，SSE2000-114，IN2000-65，CS2000-45(2000)

[2] 浦西，神田，島村；パケットアセンブリによるトラヒックへの影響に関する一考察，電子情報通信学会ソサイエティ大会，B-11-19，一般講演 (2000)

[3] 浦西，神田，島村；パケットアセンブリにおけるプライオリティキューイングを利用した実時間トラヒックの性能向上，電子情報通信学会総合大会，一般講演 (2001)

[4] 浦西，神田，島村；インターネットトラヒックを考慮したパケットアセンブリ転送方式の効率化に関する検討，信学技法，NS2001-237，IN2001-193(2002-03)

[5] JPIX のトラヒック推移　http://www.jpix.co.jp/jp/techncal/traffic.html

[6] ルーティングテーブルの経路制御情報の現状　http://bgp.potaroo.net/

[7] MCI パケットサイズ分布　http://www.vbns.net:8080/presentations/papers/MCItraffic.pdf

[8] 神田，浦西，島村；パケットアセンブリによるルータ負荷への影響，電子情報通信学会全国大会，一般講演（2003-03）

[9] What is QoS?　http://www.sitaranetworks.co.jp/qos/qos_4.html

[10] NSPIXP2 Traffic http://nspixp.sfc.wide.ad.jp/Traffic/

[11] http://www.juniper.net/news/features/ipii/faq_ip2.html

[12] 堀内，浅谷；高効率パケット多重化転送ネットワークの検討，信学技法，CQ2000-38

[13] 雲翔，浅谷；パケット一括による高スループット転送方式の検討，信学技法，NS2001-236，IN2001-192

[14] 篠宮，中川，山岡，マルチメディアストリームの特徴を考慮したパケットマージ手法の検討，信学技法，NS2002-131，IN2002-86

[15] 野村；マルチパケット伝送方式の評価，研究実用化報告第 31 巻第 12 号 (1982)

[16] 中川，江崎，菊池，永見；光スイッチを用いた次世代インターネットエクスチェンジの設計，電子情報通信学会誌 Vol.85 No.5 pp.363-369 2002 年 5 月

[17] 浦西，島村；ネットワークトランスファ性能評価とその高機能化に関する研究，学士学位論文

[18] 小高；基礎からわかる TCP/IP アナライザ作成とパケット解析，ISBN 4-274-06404-2

[19] 谷岡，島村；パケットアセンブリ装置の性能評価とその高機能化に関する研究

[20] 山田，島村；パケットアセンブリによるエッジルータ負荷に関する研究

[21] 竹下，村山，荒井，苅田；マスタリング TCPIP 入門編　第 2 版，ISBN 4-274-06257-0

[22] Ivan Pepelnjak，Jim Guichard；MPLS&VPN アーキテクチャ　ISBN4-7973-1611-x

[23] OPNET http://www.opnet.com/

[24] 戸田；ネットワーク QoS 技術，ISBN4-274-07921-x

[25] 長；インターネットにおける QoS 制御技術
http://www.nic.ad.jp/ja/materials/iw/1999/notes/C8.PDF

[26] 帯域測定ツール　NetMi ver.1.12
http://www02.so-net.ne.jp/ fyoshi/soft/index.html

付録 A

予備実験

A.0.3　アセンブリ待機時間による影響

RTT が小さい場合のアセンブリによる TCP フローへの影響をシミュレーションで示す．図 A.1 のモデルでシミュレーションを行った．フローへの影響を調べたシミュレーションと同様のモデルであるが，RTT を 30ms から 8.37ms と小さくしている． 受信バッファサ

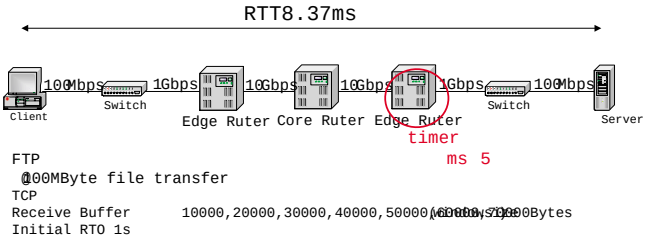

図 A.1　TCP フローへの影響のシミュレーションモデル

イズ（ウインドウサイズ）を大きくすれば，次々にパケットが転送することが可能となり（極端には Ack を待たずに転送すること），アセンブリの影響は小さくなるのではないかという予想があったが，サイズを大きくしてもフローへの影響は変わらなかった．

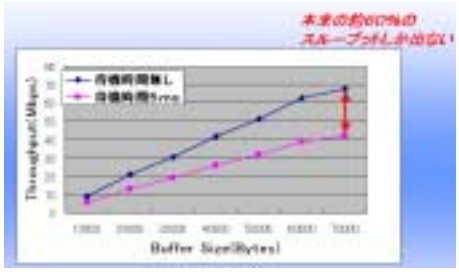

図 A.2　アセンブリタイムアウトによる TCP スループットの影響

付録 B

パケットキャプチャについて

パケット転送間隔を調べる実験を行なった．帯域測定ツール NetMi ver.1.12[26] を用いて, 図 3.9 のようなモデルで,PC から summit48 のスイッチに向けて,UDP を用いてパケットサイズ 1514Byte で転送を行なった. パケット間隔の測定には Ethereal を用いた.

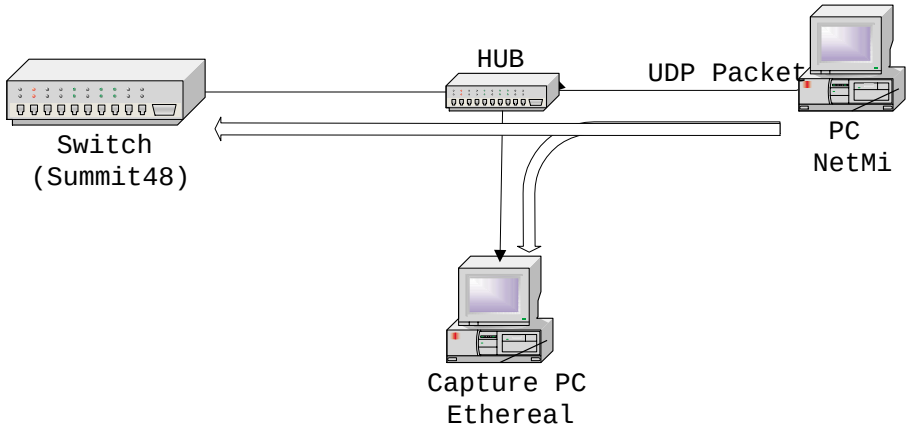

図 B.1　パケットキャプチャモデル

Ethereal によるパケットキャプチャの結果で，パケット到着間隔について調べるとその値が 17 μ s であった．これは

$$\frac{1500Byte \times 8bit}{100Mbps} = 0.00012 = 120\ \mu\ s$$

100Mbps の計算の 120 μ s とは大きく離れている.

この原因として考えられることに，パケットのキャプチャの方法に問題がある。Ethernet インタフェースには,64KByte 程度の FIFO バッファが用意されている. トラヒック調査での理想としては，パケットが到着するたびにその到着時間をアプリケーションで記録でき

ればよい．しかし，パケットが到着するたびにパケットをアプリケーションへ渡すことは，PC がそれだけの処理を行うわけではないことを考えると，効率化のために，パケットを受信しても、すぐにアプリケーションにパケットが渡されず（時間が記録されず)、ある程度パケットがバッファに蓄積されてからまとめてからアプリケーションに渡されるということが考えられる．

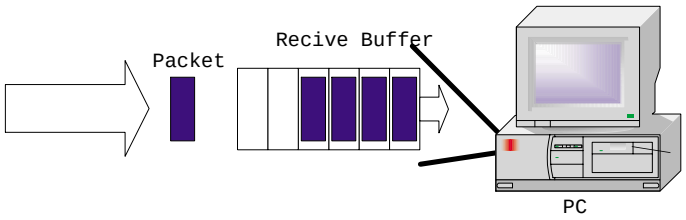

図 B.2　バッファへのパケット蓄積

これは，アプリケーションで処理できる状態になるまでの時間というように考えることができる．よって，測定結果のパケット間隔は、バッファからの複数のパケットをリードする速度になり，パケットがその間隔でノードに到着したという意味とは異なる．そのために，100Mbps の速度では考えられない 17 μ s という短い到着間隔が表示される結果になったと推測される．パケット到着間隔は，パケットアセンブリとって重要な値であるため，今後このパケットキャプチャについて調査する必要がある．

B.1　実際にコアルータに負荷がかかっている場合のシミュレーション

複数のトラヒックを発生させ，コアルータがボトルネックになっている状態をシミュレーションし，そのボトルネックをパケットアセンブリを行うことにより解消するシミュレーションを行った．

B.1.1　シミュレーションモデル

シミュレーションには，図 B.3 のようなモデルを用いた．

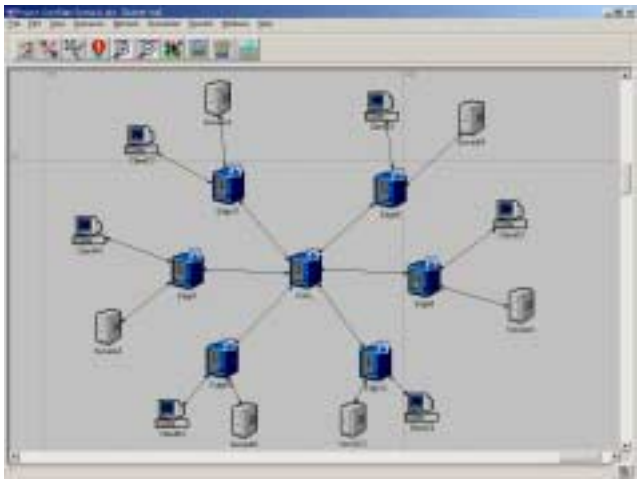

図 B.3　シミュレーションモデル

コアルータを中心にそれぞれのエッジルータが接続され，そのエッジルータには FTP のクライアントとサーバが一台ずつ接続されている．これらの端末がコアルータを介して FTP 転送を行うようにシミュレーションの設定を行った．ボトルネックのシミュレーションとは異なり，エッジルータとコアルータの性能 (packet per second) は同じである．

B.1.2　結果

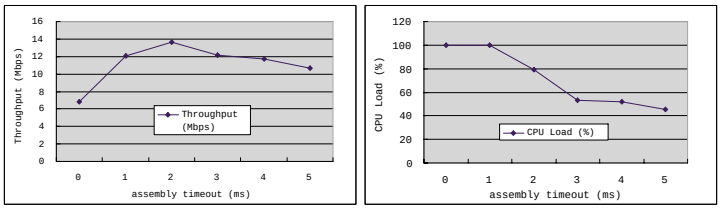

図 B.4　シミュレーション結果

アセンブリのタイムアウトを大きくすることで多くのパケットをアセンブリしすぎることは，それぞれのフローのスループットを低下させることになり，またコアルータの処理能力を無駄に使うことになるという，コアルータをボトルネック（2500pps）に設定した時と同様の結果が得られた．