特別研究報告

題目

TCP プロキシ機構の実ネットワーク上での性能評価

指導教員

中野 博隆 教授

報告者

山根木 果奈

2005 年 2 月 17 日

大阪大学 基礎工学部 情報科学科

平成 16 年度 特別研究報告

TCP プロキシ機構の実ネットワーク上での性能評価

山根木 果奈

内容梗概

近年、ユーザの要求するサービスは多種多様で高度になっている。この要求に応えるため、我々はアプリケーション層や IP 層で品質制御を行うのではなく、トランスポート層において品質制御を行う TCP オーバレイネットワークに関する研究を行っている。TCP オーバレイネットワークを構築する上で基本的な技術要素となる TCP プロキシ機構は、通常データ転送のために設定されるエンド端末間の TCP コネクションを、ネットワーク内のノードにおいて分割し、複数の分割 TCP コネクションを用いて中継データ転送を行う機構である。TCP プロキシ機構を用いることで、TCP コネクションの見かけのフィードバックループが小さくなるため、データ転送スループットが向上することが期待される。本報告では、大阪大学と NEC を結ぶインターネット公衆回線を用いたデータ転送実験を通じて TCP プロキシ機構の性能評価を行う。その結果、従来の TCP によるデータ転送では十分なスループットが得られない状況においても、エンド端末の TCP の輻輳制御機構のパラメータやアルゴリズムを変更することなく、従来方式に比べて最大約 4 倍のスループットが得られることを明らかにしている。また、TCP プロキシ間の TCP コネクションにおいて高速データ転送を可能とする輻輳制御方式として HSTCP および gHSTCP を用いた場合、最大スループットが約 5.5 倍向上することを示している。

主な用語

TCP (Transmission Control Protocol), 実装, TCP プロキシ, High-Speed TCP

目 次

図 目 次

表 目 次

1 はじめに

ADSL (Asymmetric Digital Subscriber Line) や FTTH (Fiber To The Home) といった広帯域アクセス網技術の進展により、近年ますますインターネットが発展し、ユーザ数の爆発的な増加にともない、要求されるサービスが多様化している。それらの中には、エンドホスト間のスループットなどに関して高いネットワーク品質を要求するサービスもあるが、現在のインターネットはベストエフォート型であるため、ユーザの要求品質を必ずしも満たすことはできない。

この問題を解決し、IP 層において品質制御を行う技術として IntServ [1] や DiffServ [2] などが存在する。例えば DiffServ においては、サービスの種類によってルータにおけるパケット処理の優先順位を決定することによって、各フローの通信品質の差別化を行うことを目的としている。しかしながら、IntServ や DiffServ によってユーザの要求品質を十分に満足するためには、フローが通過するすべてのルータに品質制御機能が実装されている必要があり、ネットワーク規模に対するスケーラビリティ、導入コストなどの面から実現は困難であると考えられる。

また、IP 層より下位あるいは上位層において品質制御を行う技術も存在し、MPLS (Multi-Protocol Label Switching) [3] や GMPLS (Generalized MPLS) [4] はアンダーレイネットワーク技術の代表例である。例えば MPLS では、制御を受けるコネクションに属するパケットはルータごとに経路決定が行われるのではなく、エッジルータですべての経路決定が行われるため、コアルータにおいて高速なパケット転送を可能としている。これらの技術は、データ転送が単一の ISP (Internet Service Provider) 内で完結する場合には効率よく機能するが、複数の ISP を経由する場合には bandwidth broker [5] などの新たな制御機構が必要である。したがって、IP 層で実現する品質制御技術と同様に、ネットワーク規模に対するスケーラビリティなどに問題がある。

CDNs (Contents Delivery Networks) [6] におけるプロキシキシキャッシュサーバや P2P (Peer to Peer) ネットワークはオーバレイネットワーク技術の代表例である。オーバレイネットワークにおいては、送信ホストから送信されたパケットはそのネットワークに属しているほかのホスト/ノードを経由して受信ホストに転送される。その際、パケットが転送される

送受信ホスト間の経路は、オーバレイネットワーク上の論理的な仮想パスから構成される。そのため、オーバレイネットワークが複数の ISP 上に構築された場合にも, 既存の IP ネットワークを変更することなく、さまざまなサービスを提供することが可能である。オーバレイネットワークにおいては、モニタリングやシグナリングによって下位の IP ネットワークから情報を取得することによって通信品質の制御を行う。例えば [7–9] に示されるようなアプリケーションオーバレイネットワークにおいては、オーバレイノード間のラウンドトリップ時間 (RTT) やホップカウントを用いることによって、オーバレイネットワークの構築や送受信ノード間の適切な経路選択を行っている。また、RON (Resilient Overlay Network) [10] や FBR (Feedback Based Routing) [11] はルーティングオーバレイネットワークの例である。これらは、オーバレイノード間の経路の品質を計測によって獲得し、その情報をもとにオーバレイノード間の経路選択を行っている。しかしながらこれらのオーバレイ技術は、ネットワーク品質の計測やオーバレイノード間で計測結果を交換するための付加トラヒックやシグナリングメッセージが必要である。また、アプリケーション層において品質制御を行う場合、各アプリケーションに特化した複雑な制御を必要とする、所望の性能を得るためのパラメータセッティング等が困難である、などの問題がある。

この問題に対して我々の研究グループでは、IP 層やアプリケーション層において品質制御を行うのではなく、IP 層においては従来のルーティングなど必要最低限の機能のみを提供し、品質制御をトランスポート層において行い、アプリケーションに対してより高品質なデータ転送サービスを提供する TCP オーバレイネットワーク [12] に関する研究を行っている。TCP オーバレイネットワークにおいては、通常エンドホスト間に設定される TCP コネクションを TCP プロキシと呼ばれるネットワーク内のノードで終端し、分割されたコネクションごとにパケットを中継しながら転送を行う (図 1)。これにより、TCP コネクションのフィードバックループを小さくすることが可能になるため、スループットの向上を期待することができる。また、TCP オーバレイネットワークを構築することによって、ネットワーク環境の違いを吸収・隠蔽することが可能になるため、要求されるサービス品質に応じた制御を行うことが可能になる。例えば、送受信ホスト間に無線ネットワークが含まれる場合、一般的には TCP コネクションのスループットは大幅に低下する。しかし、無線ネット

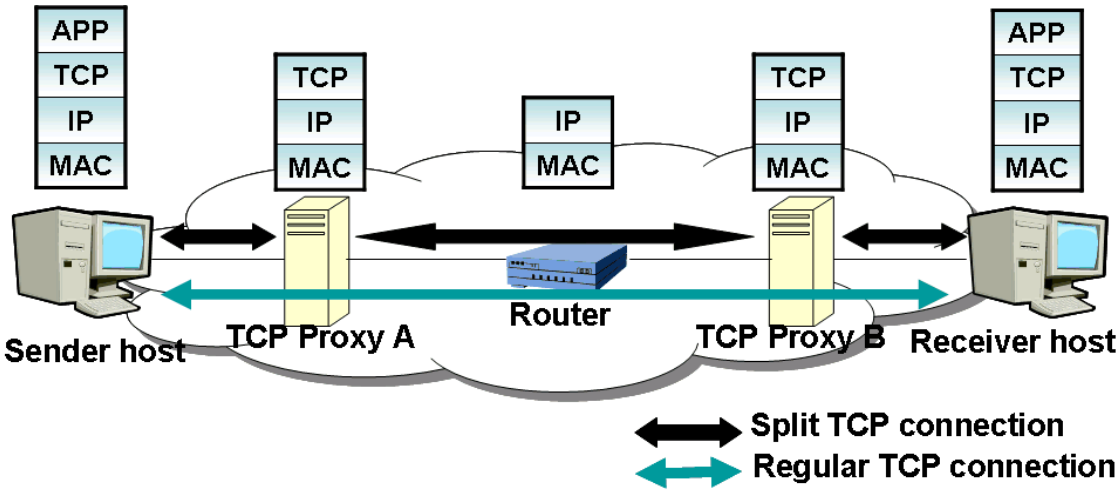


図 1: TCP オーバレイネットワーク

ワーク部分でデータ転送が独立するようにその前後でコネクション分割を行うことにより、性能劣化を最小限に抑えることが可能である。

また、アプリケーションは下位の TCP オーバレイネットワークを利用することによって以下のような恩恵を享受することができる。まず第一に、アプリケーションは TCP プロキシ間に設定された TCP コネクションから得られる情報を用いることができるため、IP ネットワークの品質の計測や経路選択といった付加的な機能をアプリケーションが持つ必要がない。第二に、TCP 層から得られる情報は IP 層から得られる情報よりもアプリケーションにとって適切である。例えば、トランスポート層の情報として得られる TCP スループットは、RON や FBR では情報が不正確であったり、情報そのものが得られない。

さらに、TCP オーバレイネットワークは高い拡張性を持つ。TCP プロキシの導入に際しては、IntServ や DiffServ のようにすべてのルータに付加的な制御機能を必要としない。また、ネットワーク内に一つの TCP プロキシしか配置されていない場合にも、様々なサービスが提供可能である。さらにネットワーク内の TCP プロキシの数が増加すれば、それにともない得られる性能はさらに大きくなる。したがって、複数の ISP を経由するデータ転送においても、TCP プロキシを段階的に導入することによって徐々にデータ転送効率が向上する。さらに、TCP プロキシがユーザの設定する TCP コネクションを自動的に分割するた

め、エンドホストのプロトコルスタックやパラメータの変更を必要としない。

TCP オーバレイネットワークを構築するための基本的な技術要素として、TCP コネクション分割機構が挙げられる。TCP コネクション分割機構は、エンドホスト間に設定される TCP コネクションを TCP プロキシにおいて複数に分割し、分割されたコネクションごとにパケットを中継しながらデータ転送を行うことを実現する機構である。我々のこれまでの研究においては、TCP コネクション分割機構による効果を、シミュレーションや数学的解析によって示してきた [13–15]。その結果、TCP コネクション分割機構を用いることにより、エンドホスト間にコネクション設定する場合と比較して平均スループットが向上すること、また、TCP プロキシが過負荷になることを防止することによって、ファイル転送遅延時間を短縮することができることが明らかになった。しかし、実ネットワークにおける TCP プロキシ機構の性能は明らかになっていない。

そこで本報告では、インターネット公衆回線を経由する実験ネットワークにおける性能評価実験を通じて、TCP オーバレイネットワークアーキテクチャおよび TCP プロキシ機構が、実ネットワークにおいても効果的であることを示す。具体的には、実コンピュータ上に実装された TCP プロキシを用い、大阪大学 (大阪府豊中市) と NEC(神奈川県川崎市) 間のネットワークを用いたデータ転送実験を行い、その性能を明らかにする。

一方、高速ネットワークにおける TCP 性能の問題を解決するために、高速 TCP と呼ばれる新たな輻輳制御手法に関する研究が近年盛んに行われている [16–20]。これらの手法は帯域遅延積が大きいネットワークにおいて高いデータ転送スループットを獲得することができる反面、送受信エンドホストの TCP を置き換える必要があるという問題点を持つ。しかし、TCP オーバレイネットワークにおいて、TCP プロキシに高速 TCP の機能を持たせることにより、送受信エンドホストの書き換えを行うことなく、高速・高遅延ネットワークにおいても高いスループットが得られると考えられる。そこで本報告では、TCP プロキシに高速 TCP を実装し、データ転送実験による性能評価を行うことで、高速 TCP の効果および TCP プロキシ機構との組み合わせによる効果を明らかにする。

本報告の構成は以下のとおりである。2 章では関連研究である TCP プロキシおよび高速 TCP の説明を行い、3 章では実験で用いるネットワークおよび実験方法を説明する。さら

に 4 章では、基礎実験により得られた実験ネットワークの基本的な性質を述べ、5 章で TCP プロキシおよび高速 TCP の効果を評価する。最後に 6 章でまとめと今後の課題を述べる。

2 関連研究

2.1 TCP コネクション分割機構

TCP コネクション分割機構は、エンドホスト間に設定される TCP コネクションをネットワーク内のノードで終端することによって TCP コネクションを複数に分割し、分割されたコネクションごとにパケットを中継しながらデータ転送を行うことを実現する機構であり、TCP オーバレイネットワークを構築するための基本的な要素技術である。TCP プロキシは、その TCP コネクションの分割およびパケットの中継転送を行うネットワーク内のノードである (図 1)。TCP プロキシを用いて行われるデータ転送においては、データの中継転送を行うための TCP コネクションが TCP プロキシ間に別途設定され、データが転送される。また TCP プロキシは、受信側ホストに代わって擬似的な ACK パケットを返信する機能を有している。これにより、送信側ホストは受信側ホストからの ACK パケットを待つことなく新たなデータパケットの送信が可能となるため、送信側 TCP が持つ見かけのラウンドトリップ時間が小さくなり、データ転送効率が向上する。さらに、コネクション分割をしている TCP プロキシは、データパケットに対する ACK パケットを受信するまでそのデータパケットをバッファリングするため、例えば TCP プロキシと受信側ホスト間でパケットが廃棄された場合に、TCP プロキシから廃棄パケットを再送することを可能にしている (図 2)。これにより、従来の TCP が持つデータ転送の信頼性を維持しつつ、効率のいいデータ転送を実現している (図 3)。

送信側ホストは、送信側ホストに一番近い TCP プロキシから擬似的な ACK を受け取り、一方受信側ホストは、送信側ホストから送信されていると見せかけられたパケットを受信側ホストに一番近い TCP プロキシから受け取る。またパケット廃棄が起こった場合は、分割されたコネクション間で再送が行われる。したがって送受信エンドホストは、TCP コネクションが TCP プロキシによって分割されていることを意識することなくデータ転送を行うことができるので、送受信エンドホストで TCP プロキシを用いるために特別な設定を行う必要がない。このことは上位レイヤのアプリケーションにも設定が必要ないことを同時に示している。

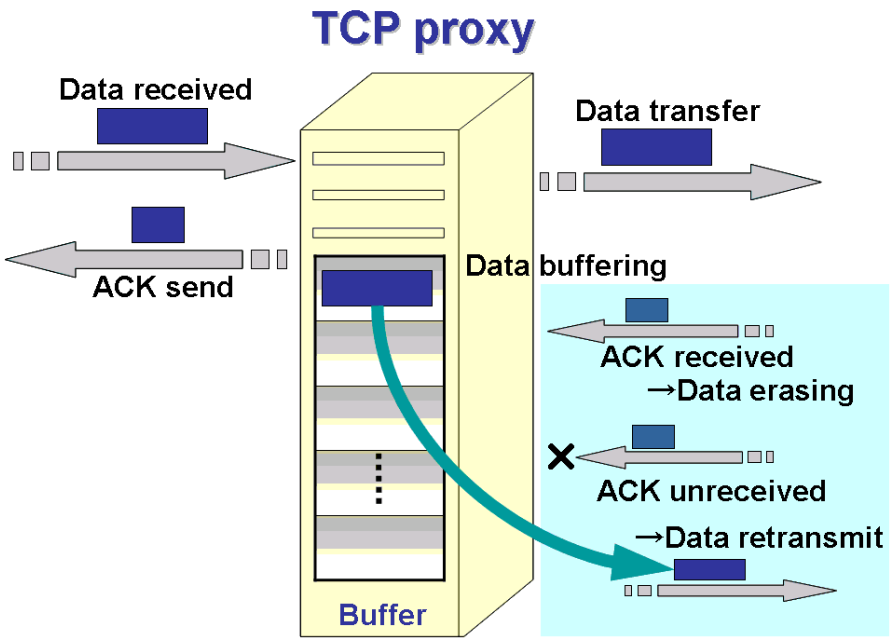


図 2: TCP プロキシの機能

2.2 高速 TCP

TCP Reno バージョンに基づいて実装されている現在の OS で用いられている TCP は、リンク帯域が 100 Mbps を超えるような高速ネットワークにおいては、十分なスループットを得ることができないことが知られている [16]。この問題に対して、TCP の輻輳制御アルゴリズムを改善することで高いスループットを獲得する、高速 TCP と呼ばれる手法が数多く提案されている [16–20]。本報告では、それらのうち文献 [16] において提案されている High Speed TCP (HSTCP)、およびその改良である Gentle High Speed TCP (gHSTCP) [19] を対象とし、TCP プロキシに実装しデータ転送実験を行うことによって、TCP プロキシに高速 TCP を適用することの有効性を検証する。

2.2.1 High Speed TCP (HSTCP)

TCP Reno が高速・高遅延ネットワークにおいて十分なスループットが得られないもっとも大きな原因は、TCP コネクションの転送速度を決定する輻輳ウィンドウサイズの増加速

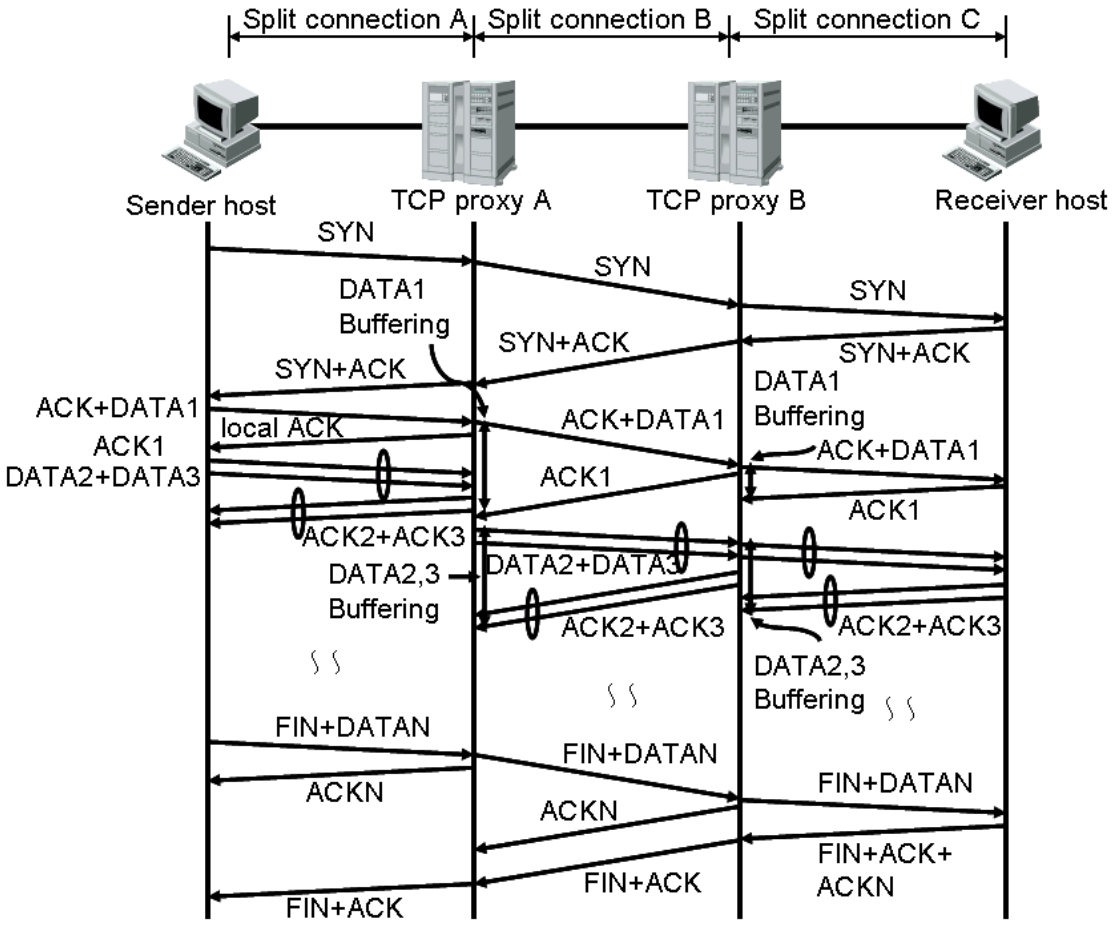


図 3: TCP コネクション分割機構

度が遅く (1 ラウンドトリップ時間あたり 1 パケット)、かつパケット廃棄を検知した際の減少幅が大きい (半減) ことである。この問題に対し、[16] において提案されている HSTCP においては、現在の輻輳ウィンドウサイズが大きい場合には、その増加速度を大きくし、減少幅を小さくする (図 4)。これにより、帯域遅延積の大きなネットワークにおいても輻輳ウィンドウサイズが大きく維持されるため、高いスループットを得ることができる。HSTCP の詳細に関しては [16] を参照されたい。

しかし、[19] において、輻輳ウィンドウサイズの増加速度が原因でパケット廃棄がバースト的に発生する、共存する TCP Reno コネクションのスループットを大幅に低下させるなどの問題点が指摘されている。本報告においては、HSTCP のスループット評価に加えて、

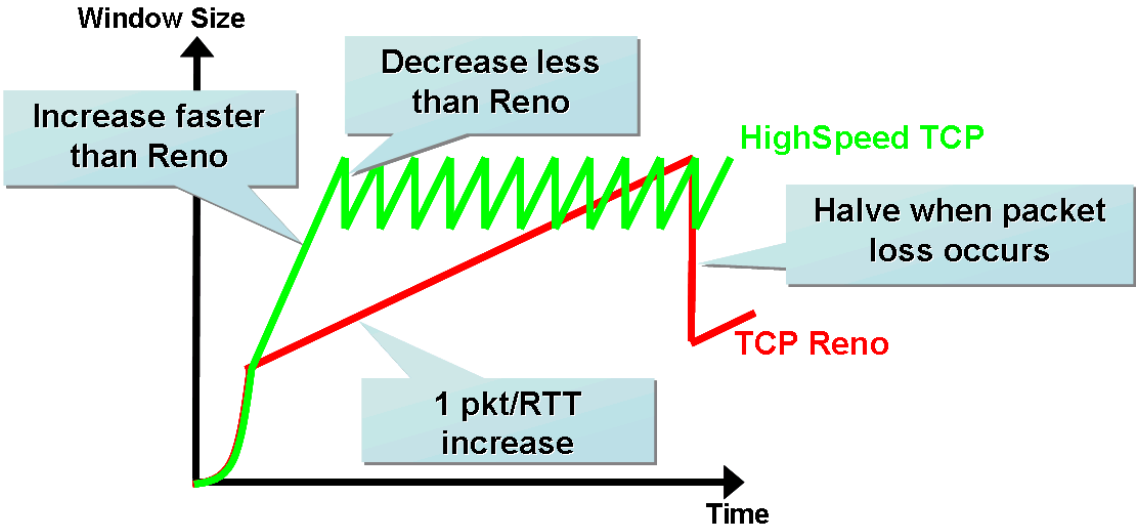


図 4: HSTCP における輻輳ウィンドウサイズの変動

共存する TCP Reno コネクションとの公平性に関する評価を行う。

2.2.2 Gentle High Speed TCP (gHSTCP)

HSTCP が持つ上述の問題点を解決するための手法として [19] において提案されている gHSTCP においては、輻輳ウィンドウサイズの増加・減少速度を自身の輻輳ウィンドウサイズの値のみに応じて変化させるのではなく、送信データパケットのラウンドトリップ時間の増加傾向を検知することによってネットワークの輻輳状態を監視し、輻輳時には TCP Reno と同じアルゴリズムを用いて輻輳ウィンドウサイズを更新する (図 5)。これにより、ルータバッファにおけるパケット廃棄率の低下や、共存する TCP Reno との公平性の向上が期待される。gHSTCP のアルゴリズムの詳細に関しては [19] を参照されたい。

gHSTCP の有効性はこれまでコンピュータ上のシミュレーションによってのみ検証が行われているため、本報告において、スループットおよび共存する TCP Reno との公平性に関して HSTCP を比較を行うことで、実ネットワークにおける gHSTCP の性能を明らかにする。

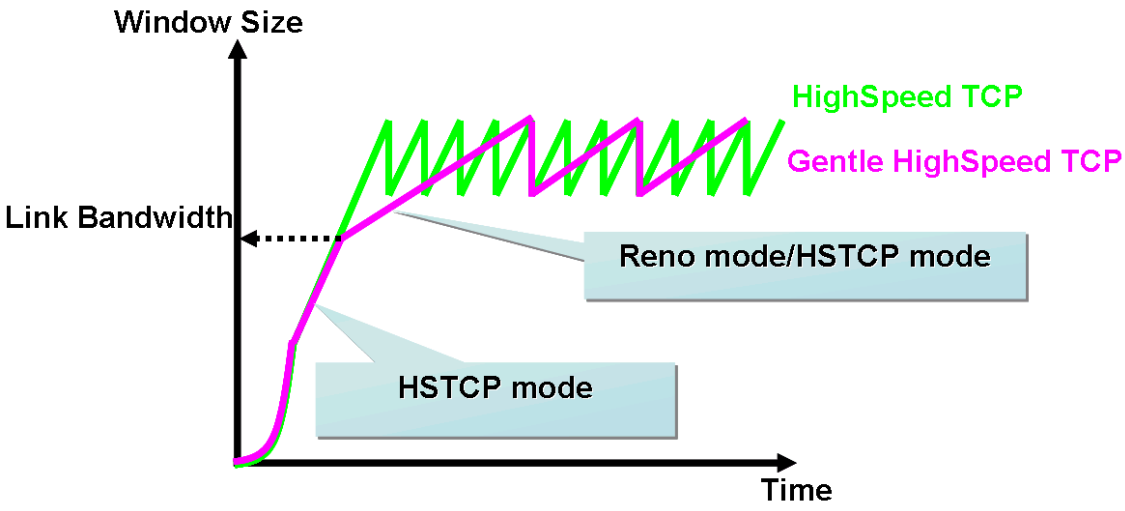


図 5: gHSTCP における輻輳ウィンドウサイズの変動

3　実験環境

本章では、本報告における実験に用いた実験ネットワーク、および実験方法の概略について述べる。

3.1　実験ネットワーク

本実験では、大阪大学 (大阪府豊中市) と NEC (神奈川県川崎市) 間に図 6 に示すような通信回線を準備し、通信実験を行う。このネットワーク環境として、ping によって測定した RTT の値、traceroute によって得たホップ数および MTU 値を表 1 に示す。また、送受信エンドホストおよび TCP プロキシが実装された端末のスペックを表 2 に示す。TCP プロキシはのプログラムは NetBSD のカーネルソースコードをベースに作成され、アプリケーション層に実装されている。プログラムはカーネルを経由せずに NIC と直接パケットのやり取りを行う。

このとき、エンドホスト間にコネクション設定され、TCP プロキシを用いない場合 (ケース 1) とエンド端末間に設定されるコネクションを TCP proxy A および TCP proxy B で分割する場合 (ケース 2) を考え、データ転送性能の比較を行う。ケース 1 では、阪大側ホスト osaka B と NEC 側ホスト tokyo B を送受信 TCP ホストとして用い、インターネットを介して通信が行われる。ケース 2 では、阪大側ホスト osaka A と NEC 側ホスト tokyo A を送受信 TCP ホストとして用い、TCP proxy A および TCP proxy B を経由して通信が行われる。したがって、ケース 2 においては、エンド間の TCP コネクションが 3 本に分割され (osaka A - TCP proxy A、TCP proxy A - TCP proxy B、および TCP proxy B - tokyo A)、TCP コネクション分割機構を用いて中継転送される。送受信エンドホストの送受信バッファはデフォルト値の 64 KBytes とし、TCP プロキシにおいてはバッファは十分大きくしている。また、NEC 側のネットワーク内にネットワークエミュレータ PacketStorm [21] を導入し、NEC と大阪大学間の伝播遅延時間を仮想的に大きくすることを可能にしている。以下では、東京-大阪間相当の通信としてネットワークエミュレータにおいて遅延を発生させない場合、および東京-沖縄間相当の通信としてネットワークエミュレータにおいて 25 msec の遅延を

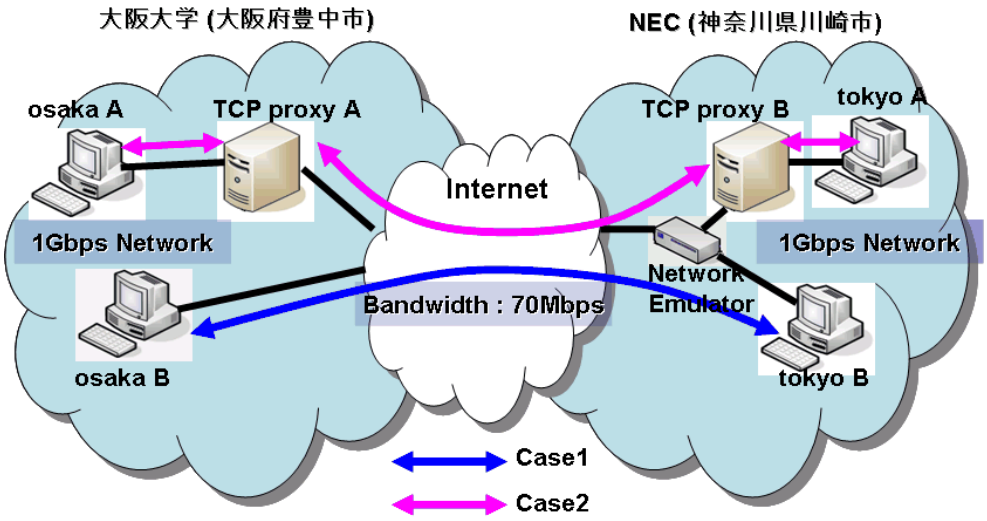


図 6: 実験環境

表 1: 実験ネットワーク環境

平均 RTT	ホップ数	MTU
16.5 msec	15	1400 Bytes

発生させる場合の実験結果を示す。さらに、SACK オプション [22] の有無がデータ転送性能に与える影響についても明らかにする。

3.2 実験方法

本実験では、帯域測定ツール iperf [23] を用いてエンドホスト間に TCP トラヒックを発生させ、受信側エンドホストでスループットを測定する。ここでスループットは、受信側エンドホストが単位時間あたりに受信するデータ量と定義する。また TCP の挙動を解析するために、ケース 1 では送信側エンドホスト、ケース 2 では送信側エンドホストに接続されている TCP プロキシの /proc/net/tcp のログを取得し、ラウンドトリップ時間 (RTT) および輻輳ウィンドウサイズ (cwnd) を測定する。さらにパケット転送、パケット廃棄および再送の様子などのパケットの細かな動きを把握するため、送信側の tcpdump [24] のデータを取得する。

表 2: 実験環境を構築する端末の性能

	送受信エンドホスト	TCP プロキシ
CPU	Intel Pentium 798 MHz	Intel Xeon 2.66 GHz
Memory	512 KB	256 KB
OS	Vine Linux 3.1	Vine Linux 3.1
kernel	2.4.26	2.4.26

また、以降の実験では、大阪大学側のエンドホスト (osaka A、osaka B) を送信側 TCP ホスト、NEC 側のエンドホスト (tokyo A、tokyo B) を受信側 TCP ホストとして用いた実験結果を示す。これは、予備実験により、大阪大学-NEC 間のネットワークにおいて、大阪大学 → NEC 向きの回線の方がスループットが高く、かつ背景トラヒックも少ないことが明らかとなったため、TCP コネクション分割機構の基本的な性質の評価に適していると考えたためである。

4　実ネットワーク環境調査

まず、本報告で用いる実験ネットワークの性質を調べるために、通常の TCP トラヒックをネットワークへ送出した場合の結果を示す。図 7 は、TCP データ転送を行った場合の、平均スループットの 2 時間における時間的変化を表している。ここでは、iperf によって TCP コネクションを 1 本、2 本、5 本、10 本、および 20 本設定し、それぞれ 2 分間のデータ転送において各コネクションの平均スループットの合計を示している。この図から、コネクション数を 5 本以上にしても合計スループットがほとんど向上しないことから、本報告で用いた実験ネットワークにおけるボトルネックリンクの帯域が約 70 Mbps であることがわかる。

図 8 は、1 本の TCP コネクションを用いたデータ転送における、TCP コネクションのラウンドトリップ時間 (RTT) と輻輳ウィンドウサイズ (cwnd) の変化をそれぞれ示したものである。この結果から、本実験ネットワークの性質について以下ように推測することができる。

- ウィンドウサイズの増減にともない RTT が増減していることから、ネットワーク輻輳によってボトルネックリンクの出力バッファにパケットが蓄積され、最終的にバッファ溢れが発生している。
- RTT の最小値が約 18 msec となっていることから、エンドホスト間の往復伝播遅延時間が 18 msec である。
- 最大の RTT が約 30 msec となっていることから、ボトルネックリンクの出力バッファが約 12 msec (リンク帯域を 70 Mbps と考えると約 100 KBytes に相当) である。
- ウィンドウサイズが約 130 パケットを超えると RTT が増加し始めていることから、このネットワークの帯域遅延積が約 160 KBytes である。
- パケット廃棄によってウィンドウサイズが小さくなる直前の RTT が安定しており、ほぼ最大 RTT に等しいことから、ボトルネックリンクの出力バッファには Tail Drop 方式が使われている。
- ほとんどの場合においてパケット廃棄時にウィンドウサイズが半減していることから、バッファ溢れによってパケットが廃棄される場合に、1 つのパケットが廃棄されている。

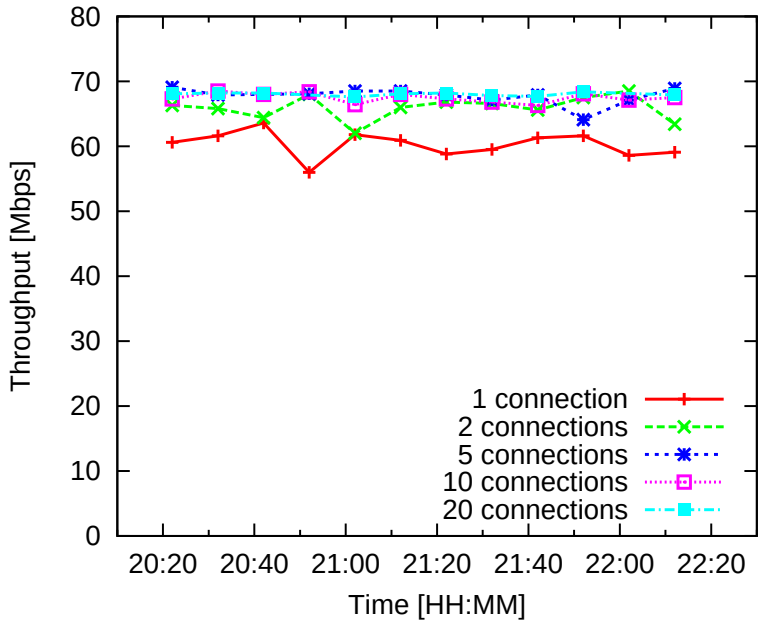


図 7: TCP スループットの時間的変動

これらの性質は、従来行われている ns-2 [25] などを用いたシミュレーションによって得られる結果とほぼ同じであるといえる。したがって、われわれがこれまでに行ってきたシミュレーションによる TCP プロキシ機構および高速 TCP の評価結果 [13–15, 19] とほぼ同様の結果が、本実験ネットワークを用いた実験においても得られると考えられる。

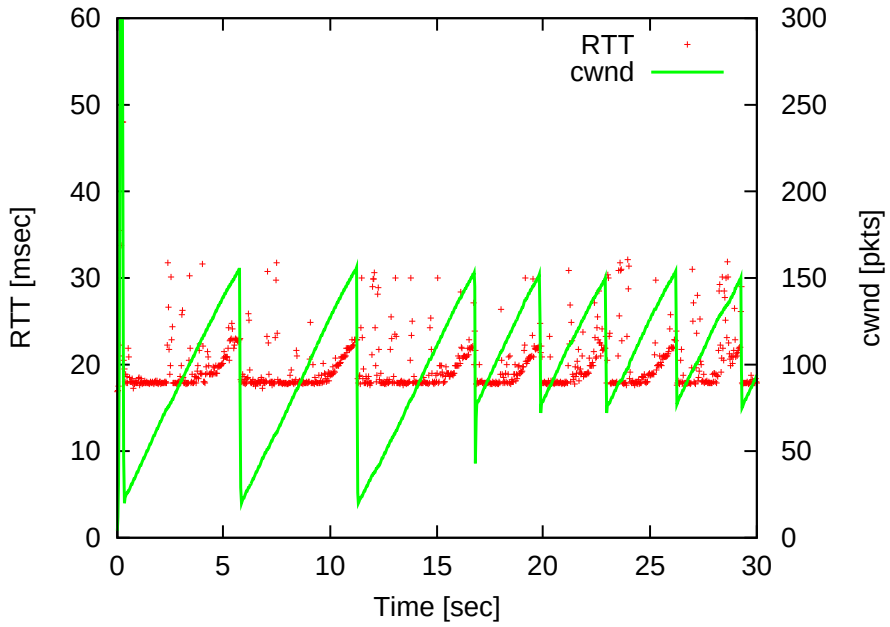


図 8: ラウンドトリップ時間と輻輳ウィンドウサイズの変動

5　実験結果と考察

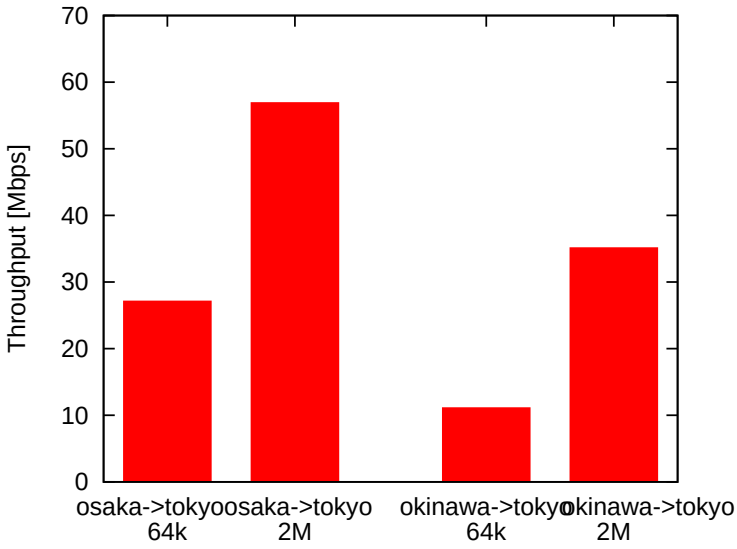
本章では、データ転送実験結果を示し考察を行うことで、TCP プロキシ機構および高速 TCP の有効性について議論する。

5.1　TCP プロキシの効果

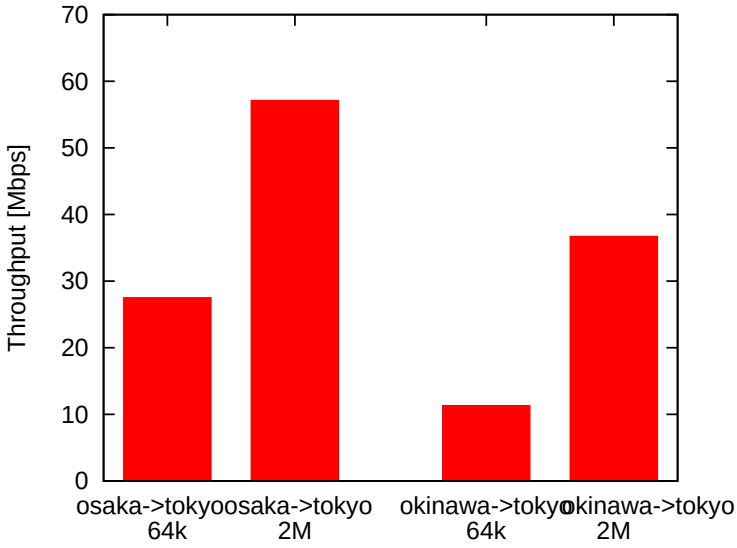
図 9 および図 10 は、それぞれ TCP プロキシ機構を用いない場合と用いる場合について、送受信エンドホストの TCP ソケットバッファサイズおよび送受信エンドホスト間の遅延時間を変化させた時の、データ転送の平均スループットを示している。ここで図 9(a) および図 10(a) は SACK オプションを用いない場合、図 9(b) および図 10(b) は SACK オプションを用いた場合を示している。実験は、2 分間のデータ転送を 5 回行い、その平均値を示している。TCP プロキシを用いない場合はケース 1 に相当し、送受信エンドホスト間に TCP コネクションを 1 本設定しデータ転送を行っている。一方 TCP プロキシを用いる場合はケース 2 に相当し、図 6 に示す 2 箇所の TCP プロキシにおいてコネクション分割を行い、3 本の中継コネクションを用いてデータ転送を行っている。

図 9 から、東京-大阪間相当の通常のデータ転送においては、TCP コネクションが用いるソケットバッファが小さい場合には、十分なスループットが得られず、リンク帯域を有効に使うためには、ソケットバッファを大きく設定する必要があることがわかる。これは、本実験で用いたネットワークの帯域遅延積は約 160 KBytes であるため、64 KBytes のソケットバッファではリンク帯域を使い切ることができないためである。また、東京-沖縄間相当の実験結果から、エンド端末間の遅延時間が大きくなると、バッファを大きくしてもリンク帯域を使い切ることができないことがわかった。これは、2 章において説明したように、TCP Reno のウィンドウサイズの制御アルゴリズムが、ネットワークが輻輳していない場合には 1RTT に 1 パケット分だけ大きくし、パケット廃棄を検知したときに半減させるため、大きな帯域遅延積を有効に使うことができないためである。

一方、TCP プロキシを導入してコネクション中継を行った場合（図 10）には、東京-大阪間相当の実験結果から、エンド端末のバッファが小さい場合においても、非常に高いスルー

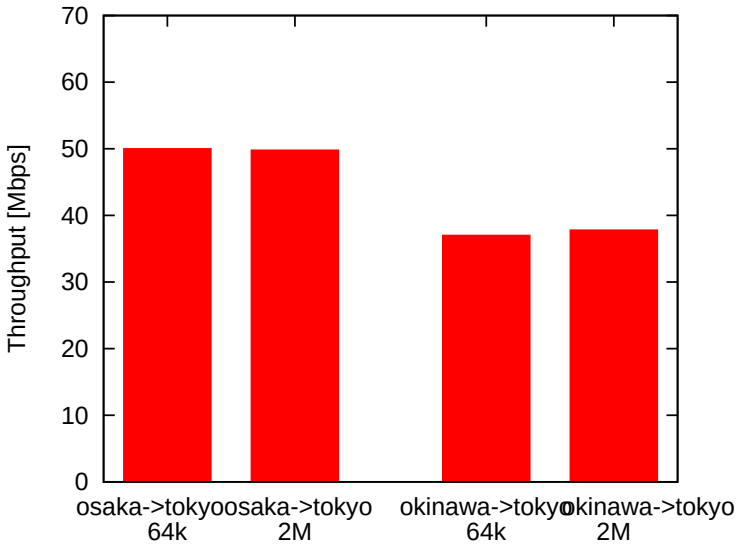


(a) SACK オプションを用いない場合

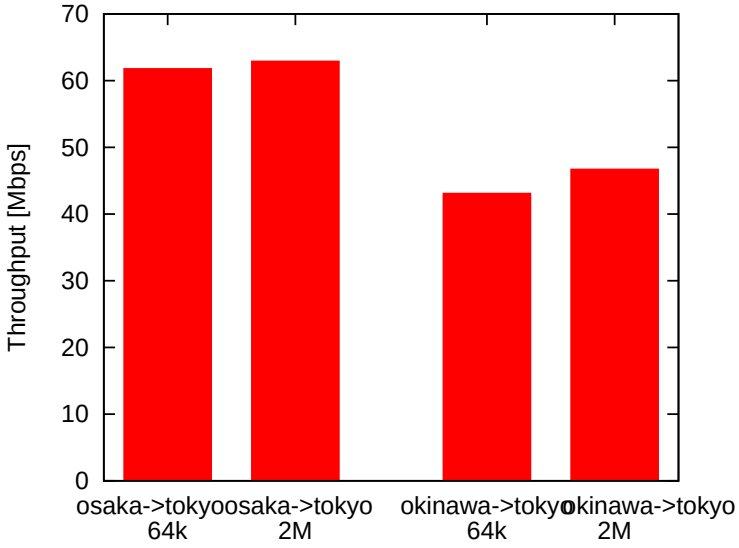


(b) SACK オプションを用いた場合

図 9: TCP プロキシを用いない場合のスループット



(a) SACK オプションを用いない場合



(b) SACK オプションを用いた場合

図 10: TCP プロキシを用いる場合のスループット

プットが得られていることがわかる。これは、送信側ネットワーク側の TCP プロキシにおいて TCP コネクションを中継し、中継を行っている TCP プロキシ間の TCP コネクションのソケットバッファサイズを十分大きくしているため、TCP プロキシ間の TCP コネクションが高いスループットでデータ転送を行うことができること、および送信側エンドホストと TCP プロキシ間の TCP コネクションは帯域遅延積が非常に小さいため、小さなソケットバッファを用いても十分高いスループットが得られることが理由である。すなわち、TCP プロキシを導入することによって、接続されたエンドホストはプロトコルやパラメータの値を変更することなく、高いスループットを得ることができる。

また、図 9 および図 10 より、SACK オプションを用いることによって、データ転送効率が向上していることが分かる。これは SACK オプションを用いることにより廃棄されたパケットが選択的に再送されるため、廃棄パケットの再送にかかる時間が短縮され、データ転送効率が向上していると考えられる。また TCP プロキシを用いる場合の方が、SACK オプションの効果が大きいのは、TCP プロキシの実装のベースとなっている NetBSD へ実装されている SACK の効果がより高いためであると考えられる。

一方、図 10 における東京-沖縄間相当の実験結果から、エンドホスト間の遅延時間が大きい場合においても、エンドホストのソケットバッファサイズを大きく設定しなくても、大きく設定した場合とほぼ同等のスループットが得られていることがわかる。しかし、東京-大阪間相当のスループットと比較すると、東京-沖縄間相当のスループットは TCP プロキシを用いた場合においても低い。これは、TCP プロキシが両エンドホストから非常に近い場所に設置されているため、TCP プロキシ間の伝播遅延時間が、エンドホスト間の伝播遅延時間とほぼ等しいことから、分割された TCP コネクションのフィードバックループが小さくならず、TCP プロキシ間の TCP コネクションが十分なスループットを得られないためである。これは、図 9 に示す実験結果のうち、東京-沖縄間相当のネットワークにおいてはソケットバッファを大きくしてもリンク帯域を使い切ることができないことからも明白である。すなわち、TCP プロキシを設置する場所によっては、TCP プロキシを設置するだけでは十分なスループットが得られないといえる。このことは、[14] においても指摘されており、TCP プロキシの効果は、エンド間の遅延時間を等しく分割するような場所にプロキシを置くこと

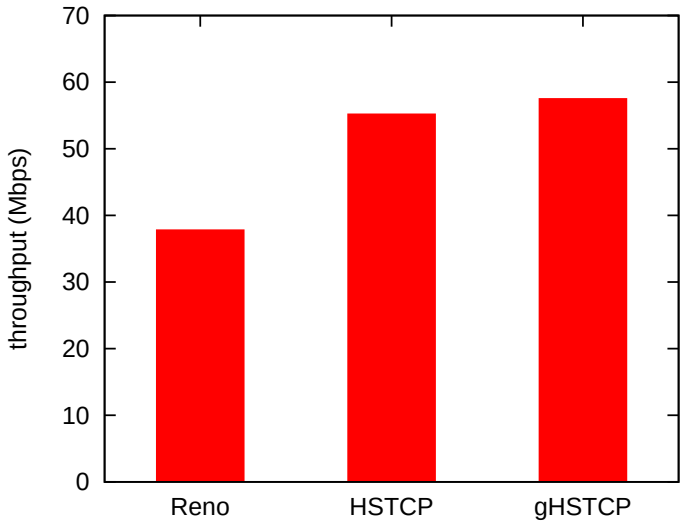
でもっとも大きくなることが示されている。

5.2 高速 TCP の効果

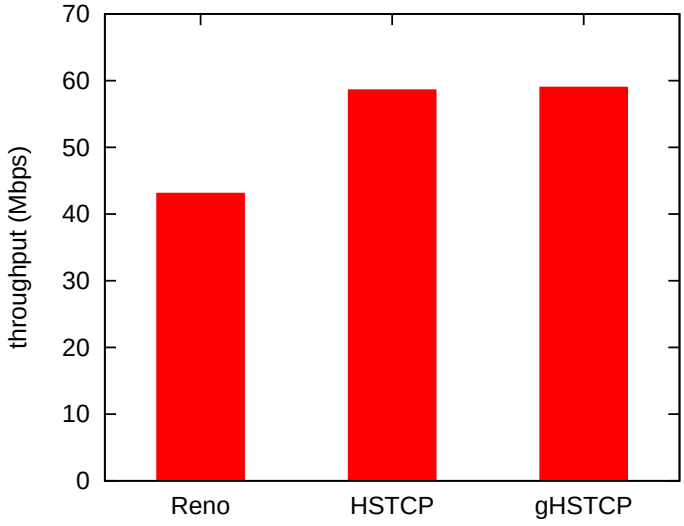
上述の問題を解決するための有効な手段の 1 つとして、TCP プロキシ間の TCP コネクションに高速 TCP を用いることが挙げられる。図 11 に、TCP プロキシ間の TCP コネクションにおいて 2.2 節で説明した HSTCP および gHSTCP を用いた場合のスループットを示す。ここで図 11(a) は SACK オプションを用いない場合、図 11(b) は SACK オプションを用いた場合を示している。この図から、TCP プロキシ間の TCP コネクションに HSTCP を用いることでエンド間のデータ転送のスループットが向上していることがわかる。また、[19] において提案した HSTCP の改良方式である gHSTCP は、HSTCP よりもさらに高いスループットを得ることができることがわかる。これは、[19] におけるシミュレーション結果に示されているように、gHSTCP を用いることによってバッファ溢れが発生するときのパケット廃棄数が大きく削減されるため、廃棄されたパケットの再送にかかる時間が HSTCP に比べて小さくなることが原因であると考えられる。これらの結果は、TCP プロキシにおいて高速 TCP を用いることで、エンド端末のプロトコルを書き換えることなく、ネットワーク環境に適した制御を可能とするトランスポート層プロトコルを用いてデータ転送を行うことができることを示している。また、SACK オプションを用いることで、特に HSTCP の場合にスループットが大きく向上していることがわかる。これは、HSTCP においてはバッファ溢れが発生する際のパケット廃棄数が多いため、SACK オプションによる再送効率の向上の効果が大きいことが原因である。

しかし、gHSTCP を用いた場合においても、得られると思われる最大スループット (約 70 Mbps) は得られていない。これは、送受信エンドホスト間の伝播遅延時間が大きいため、パケット廃棄時にタイムアウトが発生する確率が高いこと、また HSTCP におけるウィンドウサイズを増減させる際のパラメータ設定が最適でないことが原因であると考えられる。

最後に、TCP プロキシを使わないコネクションと、高速 TCP を用いる TCP プロキシを使うコネクションが同時にネットワーク内に存在した場合の、両コネクション間の公平性およびスループットに関する評価を行う。図 6 における 2 本のコネクション Case1 およ



(a) SACK オプションを用いない場合



(b) SACK オプションを用いた場合

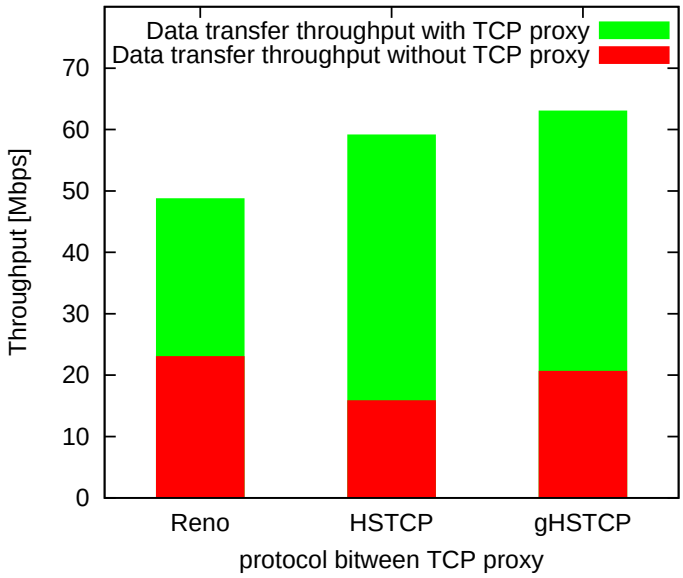
図 11: TCP プロキシ間に高速 TCP を用いた場合のスループット

び Case2 に同時にトラヒックを流し、データ転送実験を行う。図 12 は、東京-沖縄間相当のネットワークにおいて、Case2 においては送受信エンドホスト間の TCP プロトコルに TCP Reno を、Case1 においては TCP プロキシ間の TCP プロトコルに TCP Reno、HSTCP、および gHSTCP を用いた場合のそれぞれのコネクションの平均スループットを表している。また図 12(a) は SACK オプションを用いない場合を、図 12(b) は SACK オプションを用いた場合を示している。

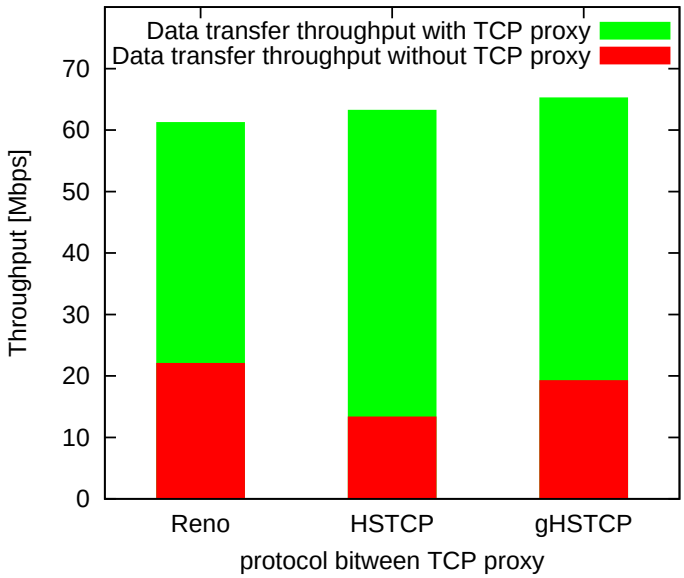
図 12(a) から、TCP プロキシ間に HSTCP を用いることで、そのコネクションのデータ転送スループットは向上するが、その反面、TCP プロキシを用いない通常の TCP コネクションのスループットを大きく低下させていることがわかる。これは、HSTCP はネットワークの輻輳状況に関係なく、自身のウィンドウサイズに応じてウィンドウサイズの増減量を調整するため、共存する TCP Reno コネクションのスループットを低下させることが原因である。

一方、高速 TCP として gHSTCP を用いた場合には、高いスループットを実現しながら、共存する TCP Reno コネクションのスループットをそれほど低下させていないことがわかる。これは 2.2.2 節で述べたように、gHSTCP はネットワークの輻輳状態に応じて自身のウィンドウサイズの増減量を調整するためである。これらの結果から、[19] において提案された gHSTCP により、共存する TCP Reno の性能を低下させることなく、自身のデータ転送スループットを大きく向上することが明らかとなった。

また図 12(b) から、SACK オプションを用いることで、TCP プロキシを用いるコネクションのスループットが向上し、TCP プロキシを用いない TCP コネクションのスループットが低下していることがわかる。これは図 9 および図 10 で示したように、TCP プロキシ間の TCP コネクションにおける SACK オプションの効果が大きいためである。このことにより、TCP プロキシ間に HSTCP を用いた場合は、通常 TCP コネクションのスループットを図 12(a) に比べてさらに低下させている。一方、TCP プロキシ間に gHSTCP を用いた場合は、TCP Reno を用いた場合と同等の通常 TCP コネクションの転送性能を保ちつつ、TCP プロキシを用いるコネクションは高いスループットを獲得できることが明らかになった。



(a) SACK オプションを用いない場合



(b) SACK オプションを用いた場合

図 12: クロストラヒックとの公平性

6　おわりに

本報告では、TCP プロキシ機構および HSTCP/gHSTCP 方式を用いることによって、それらの有効性を大阪大学 (大阪府豊中市) と NEC (神奈川県川崎市) 間の公衆網を経由する実験ネットワークにおけるデータ転送実験を通じて検証した。その結果、TCP プロキシを用いることで、エンドホストの TCP の輻輳制御アルゴリズムやソケットバッファサイズを変更することなく、高いスループットを得ることができることを示した。また、TCP プロキシ間のデータ転送プロトコルとして HSTCP を用いることで、ネットワーク遅延が非常に大きい場合など、従来の TCP Reno では十分なスループットを得ることができない状況においても高いスループットを得ることができる反面、パケット廃棄数や既存の TCP Reno コネクションとの公平性に関して問題が存在すること、および [19] で提案された gHSTCP 方式がそれらの問題点を解決し、HSTCP に比べて高いスループットと公平性を同時に満たすことができることを明らかにした。

今後の課題としては、3 地点以上のネットワークを用いたさらに大規模な公衆網を用いた実験ネットワークにおける検証、gHSTCP 方式のアルゴリズムのパラメータ調整などが挙げられる。また、100 Mbps 以上の高速ネットワークを用いた実験を行い、TCP プロキシ機構の高速ネットワークにおける性能評価をあわせて行いたい。

謝辞

本報告を終えるにあたりまして、御指導、御教授を頂きました中野博隆教授、村田正幸教授に深く感謝致します。また本報告の作成にあたり、終始熱心に指導および助言をして頂きました長谷川剛助教授に深く感謝致します。並びに的確な助言を頂きました大阪大学大学院情報科学研究科若宮直紀助教授、牧一之進助手、大阪大学大学院経済学研究科荒川伸一助手に心から感謝致します。また TCP プロキシ機構の評価に関し、様々な助言を頂きました NEC システムプラットフォーム研究所の村瀬勉様、下西英之様、浜崇之様に心から感謝致します。最後に、日頃から相談に答えて頂きました中野研究室および村田研究室の皆様方に心より御礼申し上げます。

参考文献

[1] J. Wroclawski, "The use of RSVP with IETF integrated services," *RFC 2210*, Sept. 1997.

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated service," *RFC 2475*, Dec. 1998.

[3] E. C. Rosen, A. Viswanathan, and R. Callon, "Multi-protocol label switching architecture," *RFC 3448*, Jan. 2001.

[4] E. Mannie, "Generalized multi-protocol label switching architecture," *IETF Internet Draft draft-ietfccamp-gmpls-architecture-07.txt*, May 2003.

[5] R. Bless and K. Wehrle, "IP multicast in differentiated services network," *IETF Internet Draft draft-bless-diffserv-multicast-00.txt*, Nov. 2000.

[6] S.Williams, M. Abrams, C. R. Standridge, G. Abdulla, and E. A. Fox, "Removal policies in network caches for World-Wide Web documents," in *Proceedings of ACM SIGCOMM'96*, Aug. 1996.

[7] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," *Technical Report UCB/CSD-01-1141 UC Berkeley*, Apr. 2001.

[8] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proceedins of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pp. 329–350, Nov. 2001.

[9] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-aware overlay

construction and server selection," in *Proceedings of IEEE INFOCOM 2002*, June 2002.

[10] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proceedings of ACM SOSP 2001*, Oct. 2001.

[11] D. Zhu and D. R. Cheriton, "Feedback based routing," *ACM SIGCOMM Computer Communication Review*, vol. 33, pp. 71–76, Jan. 2003.

[12] 村瀬 勉, 下西 英之, 長谷川 洋平, "TCP オーバレイネットワークの提案," **電子情報通信学会 通信ソサイエティ大会報告** (B-7-49), Sept. 2002.

[13] 牧 一之進, 長谷川 剛, 村田 正幸, 村瀬 勉, "TCP オーバレイネットワークにおける TCP コネクション分割機構の性能解析," **電子情報通信学会 技術研究報告** (IN03-198), Feb. 2004.

[14] 牧 一之進, 長谷川 剛, 村田 正幸, 村瀬 勉, "TCP オーバレイネットワークの性能解析および評価," **電子情報通信学会技術研究報告** (IN2004-96), Oct. 2004.

[15] I. Maki, G. Hasegawa, M. Murata, and T. Murase, "Throughput analysis of TCP proxy mechanism," in *Proceedins of ATNAC 2004*, Dec. 2004.

[16] S. Floyd, "Highspeed TCP for large congestion windows," *RFC 3649*, Dec. 2003.

[17] C. Jin, D. X. Wei, and S. H. Low, "FAST TCP: Motivation, architecture, algorithms, performance," in *Proceedings of IEEE INFOCOM 2004*, Mar. 2004.

[18] T. Kelly, "Scalable TCP: Improving performance in highspeed wide area networks," in *Proceedings of PFLDnet'03: workshop for the purposes of discussion*, Feb. 2003.

[19] Z. Zhang, G. Hasegawa, and M. Murata, "Analysis and improvement of HighSpeed TCP with TailDrop/RED routers," in *Proceedings of MASCOTS 2004*, Oct. 2004.

[20] T. Iguchi, G. Hasegawa, and M. Murata, "A new congestion control mechanism of TCP with inline network measurement," in *Proceedings of ICOIN2005*, Jan. 2005.

[21] PacketStorm Communications, Inc., "PacketStorm Communications, Inc. - Providing the Internet in a Box," available from `http://www.packetstorm.com/`.

[22] M. Mathis, "TCP selective acknowledgement options," *Request for Comments 2018*, Oct. 1996.

[23] NARANR, "NARANR/DAST: Iperf 1.7.0 - The TCP/UDP bandwidth measurement tool," available from `http://dast.nlanr.net/Projects/Iperf/`.

[24] LBNL's Network Research Group, "TCPDUMP public respository," available from `http://www.tcpdump.org/`.

[25] T. V. Project, "UCB/LBNL/VINT network simulator - ns (version 2)," available from `http://www.isi.edu/nsnam/ns/.`