特別研究報告

題目

TCP プロキシ機構のネットワークプロセッサ上での実装と評価

指導教官

村田正幸 教授

報告者

松浦陽亮

平成 15 年 2 月 19 日

大阪大学 基礎工学部 情報科学科

平成 14 年度 特別研究報告

TCP プロキシ機構のネットワークプロセッサ上での実装と評価

松浦陽亮

内容梗概

近年のインターネットの発展、ユーザの爆発的な増加にともない、インターネット上で提供されるサービスは多様化している。それらの中には、エンド端末間のスループット、データ転送遅延時間、パケット廃棄率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り、ネットワーク内での通信品質が保証されないため、送受信エンド端末間で TCP (Transmission Control Procotol) やアプリケーションによる通信品質の制御を行っても、アプリケーションの通信要求を完全に満たすことはできない。この問題を解決するための IP 層における品質制御技術として、IntServ や DiffServ が挙げられるが、スケーラビリティ、導入コストなどに多くの問題を抱えている。一方、アプリケーション層における品質制御技術に関しても多くの研究が行われ、実ネットワークにおいて運用が行われているものもある。しかし、これらの技術はそれぞれのアプリケーションに特化した制御が必要であり、所望の性能を得るためのパラメータ設定が難しいという問題がある。そこで我々の研究グループでは、IP ネットワークにおいてはルーティング、パケット到達性などの必要最低限の機能のみを提供し、品質制御はトランスポート層において行う、レイヤ 4 オーバーレイネットワークに関する研究を行っている。レイヤ 4 オーバーレイネットワークにおいては、エンド端末間に設定される TCP コネクションをネットワーク内で複数のコネクションに分割し、それぞれのコネクションにおいてネットワーク環境や要求されるサービスに応じた制御を行うことで、エンド端末間の通信品質を向上することが期待できる。

本報告では、レイヤ 4 オーバーレイネットワークにおいて必要となる基本的な機能である、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・中継する機能 (TCP プロ

キシ機構) をネットワークプロセッサシステム上に実装し、その性能や処理遅延に関する評価を行う。評価に際しては、まず TCP コネクションの分割・中継に必要な処理を示し、各処理に必要な時間および処理速度を、実システムにおける計測と、解析的手法によって導出する。実装にはインテル社製のネットワークプロセッサである IXP1200 の評価ボードを用いている。評価の結果、TCP コネクションの分割・中継処理においてもっとも大きな性能上のボトルネックになるのはメモリアクセス処理であることを明らかにする。また、計測結果と推測結果の比較を行うことによって解析手法の妥当性を評価する。さらに、PC ルータ上の実装とネットワークプロセッサ上の実装の性能比較を行い、ネットワークプロセッサを用いることの有用性や特徴、および将来の高速ネットワークプロセッサシステムにおける TCP プロキシ機構の性能などに関する議論を行う。その結果、現状では PC ルータを用いた実装に性能面で劣るが、今後登場すると考えられる高速なネットワークプロセッサシステム上に TCP プロキシ機構を実装することによって、十分高速な TCP プロキシ機構を提供することが可能であることを明らかにする。

主な用語

レイヤ 4 オーバレイネットワーク、TCP (Transmission Contorol Protocol) 、TCP プロキシ機構、ネットワークプロセッサ、IXP1200

目 次

図 目 次

表 目 次

1 はじめに

近年のインターネットの発達、ユーザの爆発的な増加にともない、インターネット上で提供されるサービスは多様化している。それらのサービスの中には、スループット、データ転送遅延時間、パケット廃棄率等に関して高い通信品質 (QoS) を要求するものもあるが、現在のインターネットにおいては、通信品質はエンド端末で動作するトランスポート層プロトコルである TCP (Transmission Control Protocol) [1] や、TFRCP (TCP-Friendly Rate Control Protocol) [2] 等のアプリケーション層プロトコルが行う制御などに依存している。しかし、インターネットはベストエフォート型ネットワークであり、ネットワーク内では通信品質は一切保証されないため、送受信エンド端末において行う品質制御だけでは、希望の通信品質を実現することは本質的に不可能である。

この問題を解決し、エンド端末間の QoS 保証を実現するために、IP 層において品質制御を行う様々な技術が研究開発されている。例えば IntServ アーキテクチャ [3] では、RSVP (Resource ReSerVation Protocol) [4] 等のプロトコルを用いることによってネットワーク資源の予約を行い、各フローが用いるネットワーク帯域、最大転送遅延等を保証することができる。しかし、IntServ においては、通過するすべてのフローの状態をネットワーク内の全ルータが保持し、その情報をシグナリングプロトコルを用いてルータ間で交換する必要があるため、ネットワーク規模に対するスケーラビリティに大きな問題を持つ。また DiffServ アーキテクチャ [5] では、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実現することができる。DiffServ は IntServ と違い、フロー情報はエッジルータで管理し、パケットにマーキングを行ってネットワーク内にパケットを送信し、ネットワーク内のコアルータではパケットのマーク情報にしたがって処理を行う。したがって、コアルータでフロー情報を管理する必要がないため、ネットワーク規模に対するスケーラビリティはある程度確保される。しかし、DiffServ を実現するためには、フローが通過するすべてのルータにその機能が実装される必要があり、導入時のコストが高いという問題がある。このように、IP 層において行う品質制御には様々な問題があるため、実ネットワークへの導入は極めて困難であると考えられている。

一方、品質制御をアプリケーション層で行う方法に関しても多くの研究が行われており、

IP 層のそれにくらべて実ネットワークへの適用が活発である。例えば Akamai [6] などが提供する CDN (Conetents Deliverly Network) におけるプロキシキャッシュサーバは、ユーザ (Web クライアント) からのドキュメント転送要求をプロキシサーバでいったん終端し、キャッシュを用いた代理応答や、ネットワーク輻輳状況、オリジンサーバの負荷に応じた適切なリクエスト転送を行うことで、ユーザの感じる応答時間を短縮している。また、サーバ間のキャッシュ/コンテンツ同期などのために発生するトラヒックを、クライアントからのリクエストによって発生するドキュメント転送よりも低い頻度で行う等の処理を行っている。しかし、それらの方式は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上、所望の性能を得るためのパラメータセッティング等が難しいという問題を持つ。

そこで我々の研究グループでは、IP ネットワークにおいては高度な品質制御を行わず、現在の IP ネットワークが提供しているルーティング、パケット到達性などの必要最低限の機能のみを提供するという前提のもと、品質制御をトランスポート層において行う、レイヤ 4 オーバレイネットワークアーキテクチャに関する研究を行っている。本アーキテクチャにおいては、通常エンド端末間に設定される TCP コネクションを、ネットワーク内のノードにおいて終端し、複数の TCP コネクションを中継してデータ転送を行う。これにより、分割されたそれぞれの TCP コネクションにおいて、コネクションが通過するネットワークの性質の違いや、提供するサービスに応じた制御を行うことが可能となるため、エンド端末間で 1 本の TCP コネクションを用いる従来手法に比べ、より高品質なデータ転送が可能となると考えられる。すなわち本アーキテクチャは、IP 層でコストの高い品質制御機能の実装を必要とせず、かつ上位アプリケーションによる特殊な制御を行うことなく様々なサービスを提供することができる。

TCP コネクションを分割し、それぞれのコネクションに応じた処理を行うことで性能向上を図る研究はこれまでにも行われている。しかし、それらは衛星回線、無線回線など、特定のネットワークを対象としており [7-10]、コネクション分割によるデータ転送効率が向上することは示されているが、分割・中継処理にかかる遅延時間、処理性能の評価は行われていない。提案アーキテクチャは一般的なネットワークにおけるコネクション分割を想定し、データ転送速度が 100 Mbps を越える高速処理が必要とされると考えられるため、TCP コ

ネクションの中継処理の性能評価は重要である。

そこで本報告では、レイヤ 4 オーバレイネットワークを構築するための最も基本的な機能である、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する機能 (TCP プロキシ機構と呼ぶ) を実装し、その性能および処理オーバヘッドの評価を行う。実装に際しては、標準的な PC ルータ上での実装に加え、近年着目されているネットワークプロセッサシステム上での実装を対象とする。ネットワークプロセッサはパケットの処理に適したハードウェア構成および命令セットを持つため、高速なパケット処理が可能である。またプログラマブルなパケット処理部を持つため、機能の拡張性に優れ、要求されるサービスに応じたパケット処理を実現させることが可能である。そのため、ネットワークプロセッサシステム上においてプロキシ機構を実装し、その性能を評価することは重要であると考えられる。

評価に際しては、まず TCP コネクションの分割・中継に必要な処理を挙げ、各処理に必要な時間および処理速度を、ベンチマークテスト結果などを用いて、解析的推測によって導出する。また、データ転送実験を行うことにより中継処理のスループット、遅延時間の計測を行い、推測結果との比較を行うことで、推測手法の妥当性、および相違点に関する議論を行う。またそれらの結果から、TCP プロキシ機構の性能に影響を与える要素を明らかにし、性能向上のための指針を示す。さらに、PC ルータ上の実装とネットワークプロセッサ上の実装の性能比較を行い、ネットワークプロセッサを用いることの有用性や特徴、および将来の高速ネットワークプロセッサシステムにおける TCP プロキシ機構の性能などに関する議論を行う。

本報告の構成は以下の通りである。2 章ではレイヤ 4 オーバレイネットワークの実現に必要な TCP プロキシ機構に関する説明を行う。3 章では、PC ルータおよびネットワークプロセッサシステム上での TCP プロキシ機構の実装方法を示す。4 章では TCP プロキシ機構に必要な処理の性能および遅延時間の推測方法を示し、PC ルータおよびネットワークプロセッサシステム上での処理性能の推測を行う。さらに 5 章においてデータ転送実験結果を示し、4 章で示した推測手法の妥当性、および TCP プロキシ機構の処理方式の性能に関する議論を行う。最後に 6 章で本報告のまとめと今後の課題について述べる。

2　TCP コネクション分割

本章では提案しているレイヤ 4 オーバレイネットワークを構築するための最も基本的な機能である、TCP プロキシ機構についての概略を示す。TCP プロキシ機構は、データ転送のためにエンド端末間に設定される TCP コネクションをネットワーク内のノードにおいて分割し、パケットを中継しながらデータ転送を実現する機構である。図 1 は、TCP プロキシ A および B によって TCP コネクションが 3 つのコネクション (分割 TCP コネクションと呼ぶ) に分割される様子を示している。TCP コネクションをネットワーク内で終端し、パケットの転送中継を行う技術は、これまでは衛星回線、無線回線など、特定のネットワークを対象として研究が行われてきた [7-12]。我々が提案するレイヤ 4 オーバレイネットワークアーキテクチャは、特定のネットワーク環境に特化した制御を行うのではなく、一般のネットワークにおいてエンド端末に提供するサービス品質を向上させるために、積極的にコネクション分割およびデータの中継転送を行うことを目的としている。

図 2 は、TCP プロキシ機構を用いず、通常の TCP コネクション C_{sr} を用いて送信側端末と受信側端末がルータ A およびルータ B を介して通信を行う時の、パケット送受信の様子を示した図である。ルータ A およびルータ B は受信したパケットをそのまま転送するため、パケットのやりとりは送受信端末間で直接行われる。したがって、図に示すように、ルータ B と受信側端末間でパケットが廃棄された時には、送信側端末でタイムアウトまたは重複 ACK パケットの受信によってパケット廃棄を検出し [13]、再送が行われる。

TCP プロキシ機構を導入することによって、送受信端末間の TCP コネクションが分割され、複数の TCP コネクションが送受信端末間に設定される。図 3 は、ルータ A およびルータ B が TCP プロキシ機構を実現するノード (TCP プロキシ A、B) として動作する時に、分割された TCP コネクション C_{sa}、C_{ab}、および C_{br} を通じたパケット送受信の様子を示している。コネクション C_{sa} を用いて送信側端末から送信されたパケットを受信した TCP プロキシ A は、そのパケットをコネクション C_{ab} へ中継転送する。同様に TCP プロキシ B はコネクション C_{br} へ受信したパケットを中継転送する。TCP プロキシ B は送信したパケットに対応する ACK パケットを受信側端末から受信すると、コネクション C_{ab} を通じて対応する ACK パケットを TCP プロキシ B へ送信する。同様に、TCP プロキシ A は TCP プロキ

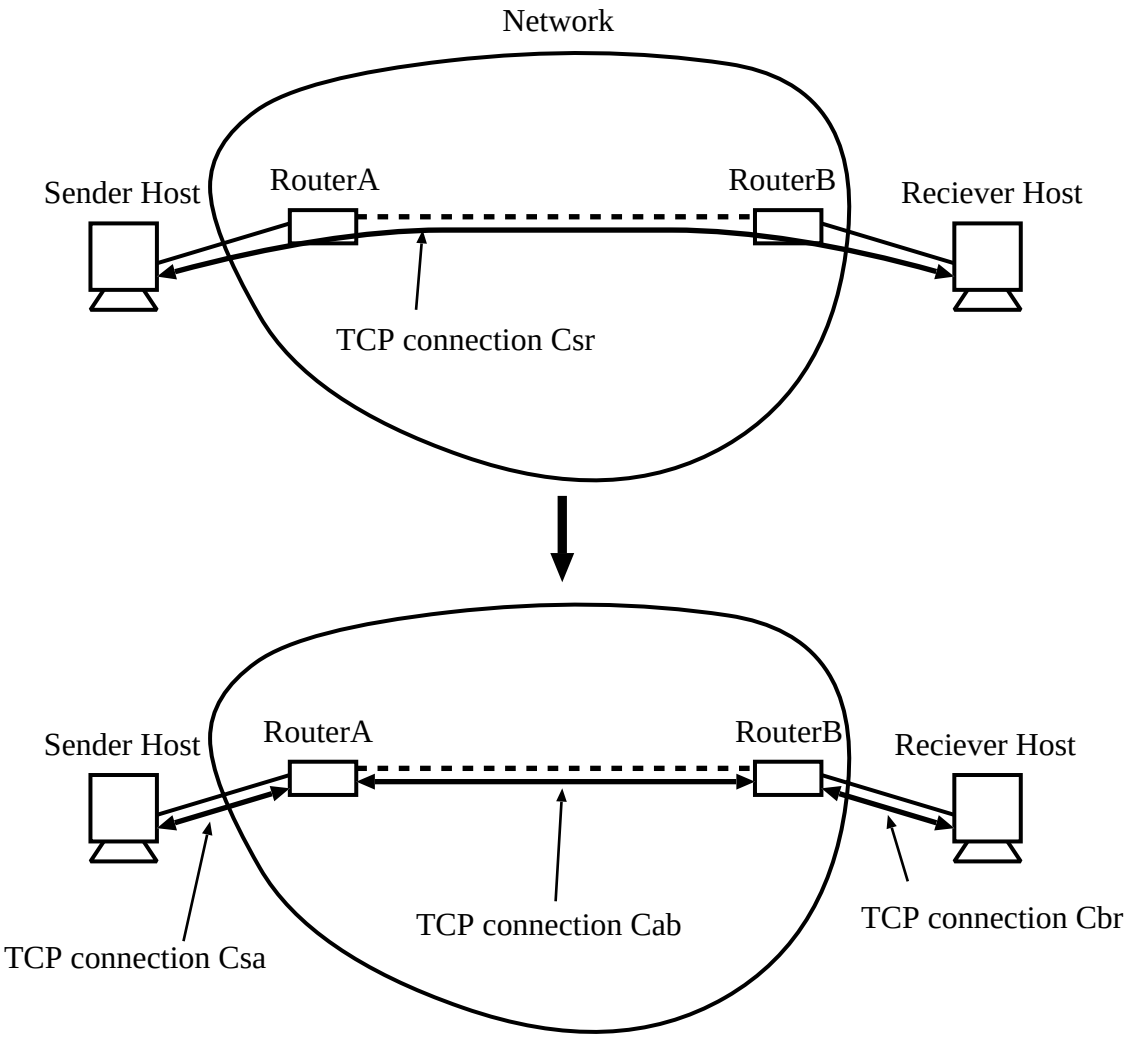


図 1: TCP プロキシ機構

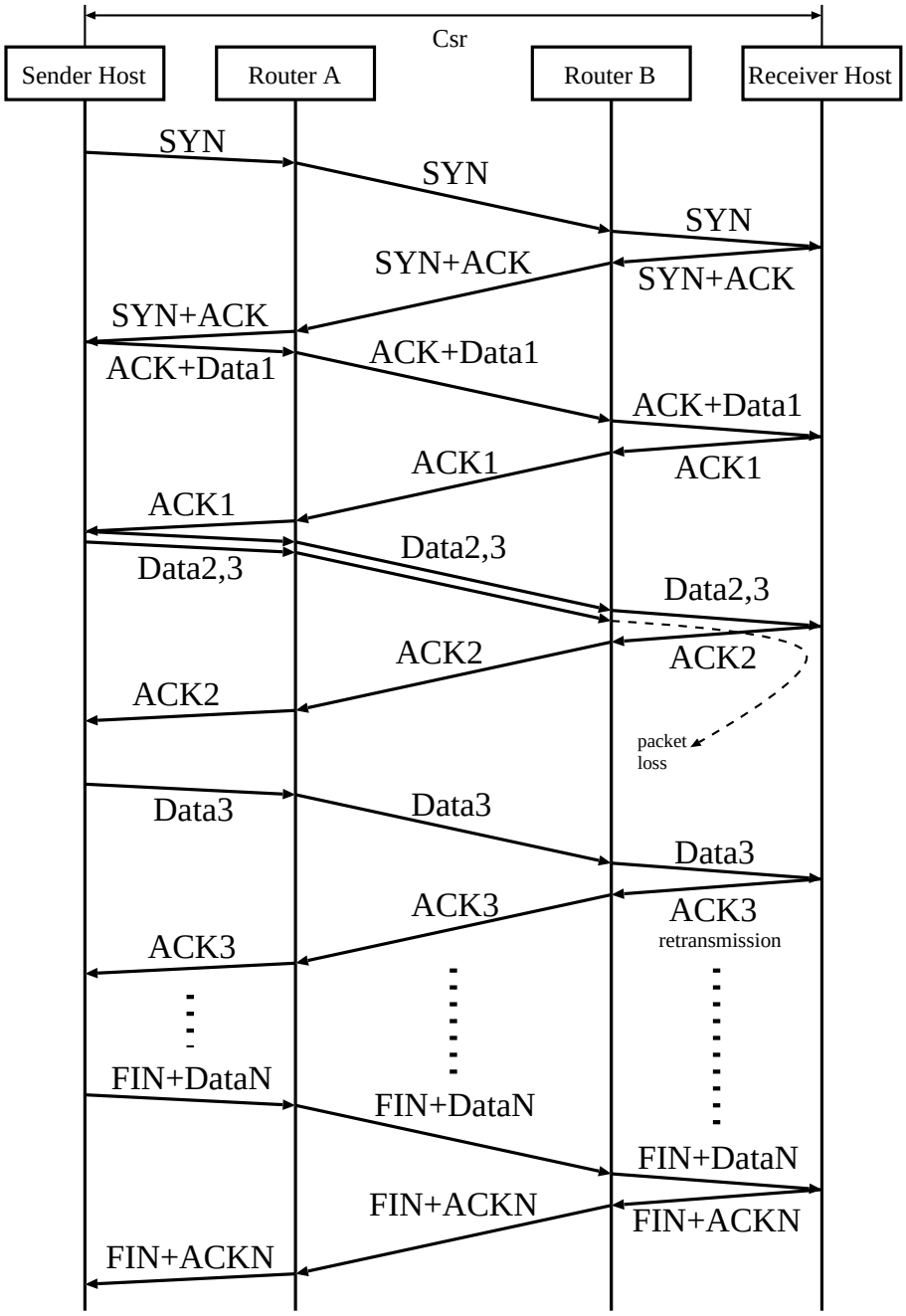


図 2: 通常の TCP コネクションを用いたパケット転送

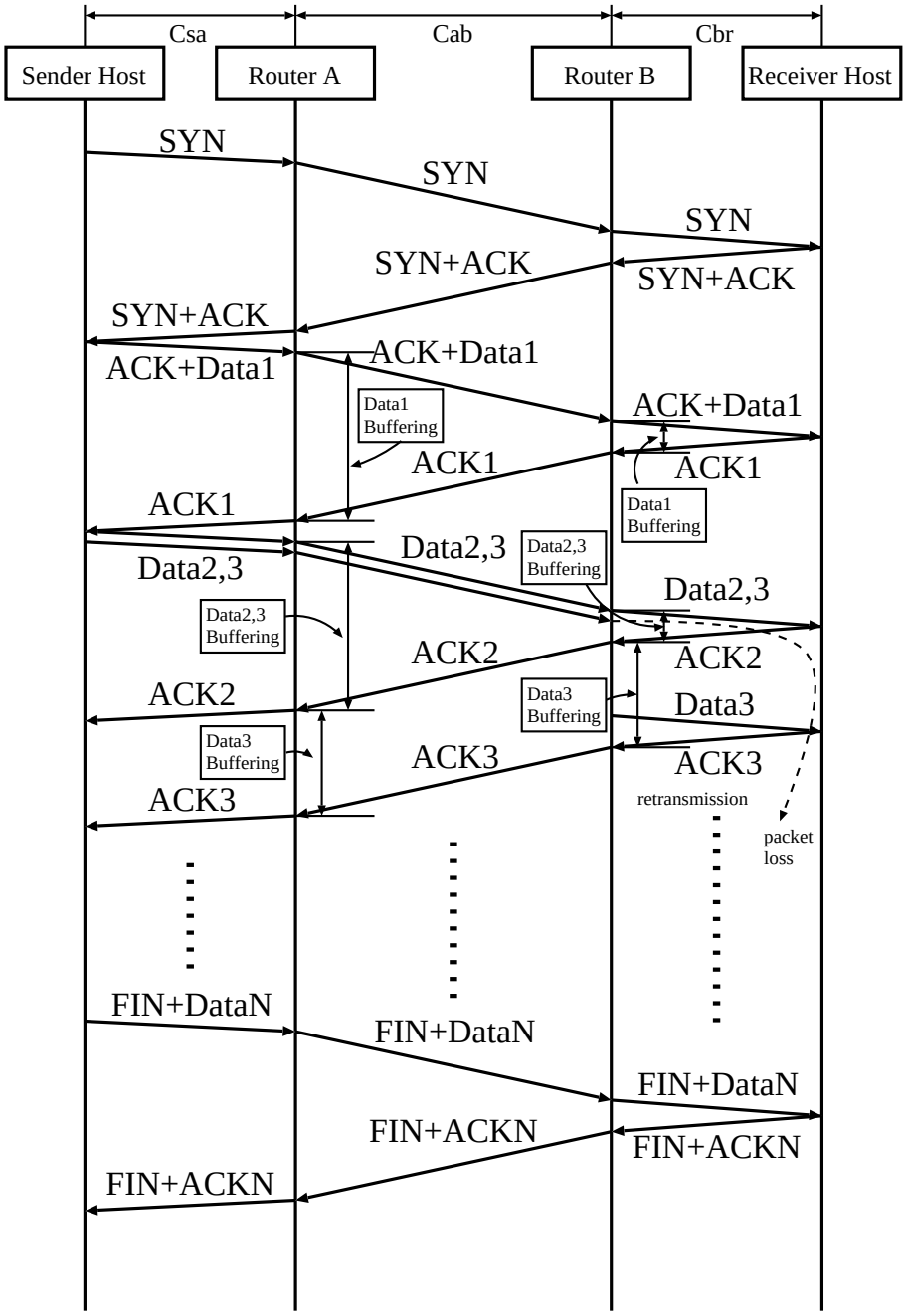


図 3: 擬似 ACK パケットを用いない TCP プロキシ機構におけるパケット転送

シ B から送信される ACK パケットを受信すると、コネクション C_{sa} を通じて対応する ACK パケットを送信側端末へ送信する。各 TCP プロキシでは、データパケットを対応する ACK パケットを受信するまでバッファリングするため、図 4 に示すように TCP プロキシ B と受信側端末間でパケットが廃棄された場合には、TCP プロキシ B 上に存在するコネクション C_{br} の送信側 TCP がそれを検出し、廃棄パケットを再送する。この局所的パケット再送によって、図 2 の場合と比べてパケット廃棄の検出および再送を素早く行うことができる。

さらに、TCP プロキシ機構はデータ転送効率を向上させる方法として、擬似 ACK パケット転送を行うことができる。図 4 は、TCP プロキシ A および B が擬似 ACK パケット転送を行う場合の、パケット送受信の様子を示している。送信側端末から転送されたデータパケットが TCP プロキシ A に到着すると、TCP プロキシ A はそれをコネクション C_{ab} を通じて TCP プロキシ B へ中継転送すると同時に、TCP プロキシ B からの ACK パケットの受信を待つことなく、擬似的に ACK パケットを生成して送信側端末へ送信する。送信側端末はその擬似 ACK パケットを受信すると、新たなデータパケットの送信を行う。これにより、送信側端末が持つラウンドトリップ時間 (RTT) が小さくなるため、図 4 に示すようにコネクション C_{sa} のデータ転送速度が向上する。同様に、コネクション C_{ab} においても TCP プロキシ B が擬似 ACK を生成して TCP プロキシ A へ送信するため、受信側端末から ACK パケットが送信されるのを待つ必要がなく、データ転送速度が向上する。

しかし、擬似 ACK パケット転送を行うと、送信側端末は受信側端末にパケットが到達する前に対応する ACK パケットを受信するため、TCP が持つ本質的な性質であるデータ転送の信頼性を損う可能性がある。例えば図 4 に示すように、TCP プロキシ B と受信側端末間でパケットが廃棄された場合、受信側端末にはパケットが到着していないにもかかわらず、送信側端末は順調にパケットが送信されていると判断してデータ転送を続けるため、送受信端末間のパケット送受信状況に大きな違いが発生する。しかし、コネクション C_{sa} を通じて TCP プロキシ A に到着したデータパケット、およびコネクション C_{ab} を通じて TCP プロキシ B に到着したデータパケットは、そのパケットの転送が完了するまでプロキシサーバ A、B においてそれぞれバッファリングされ、確実に転送処理が行われる。すなわち、各プロキシサーバにおいて中継されたパケットは、それぞれの分割 TCP コネクションにおいて

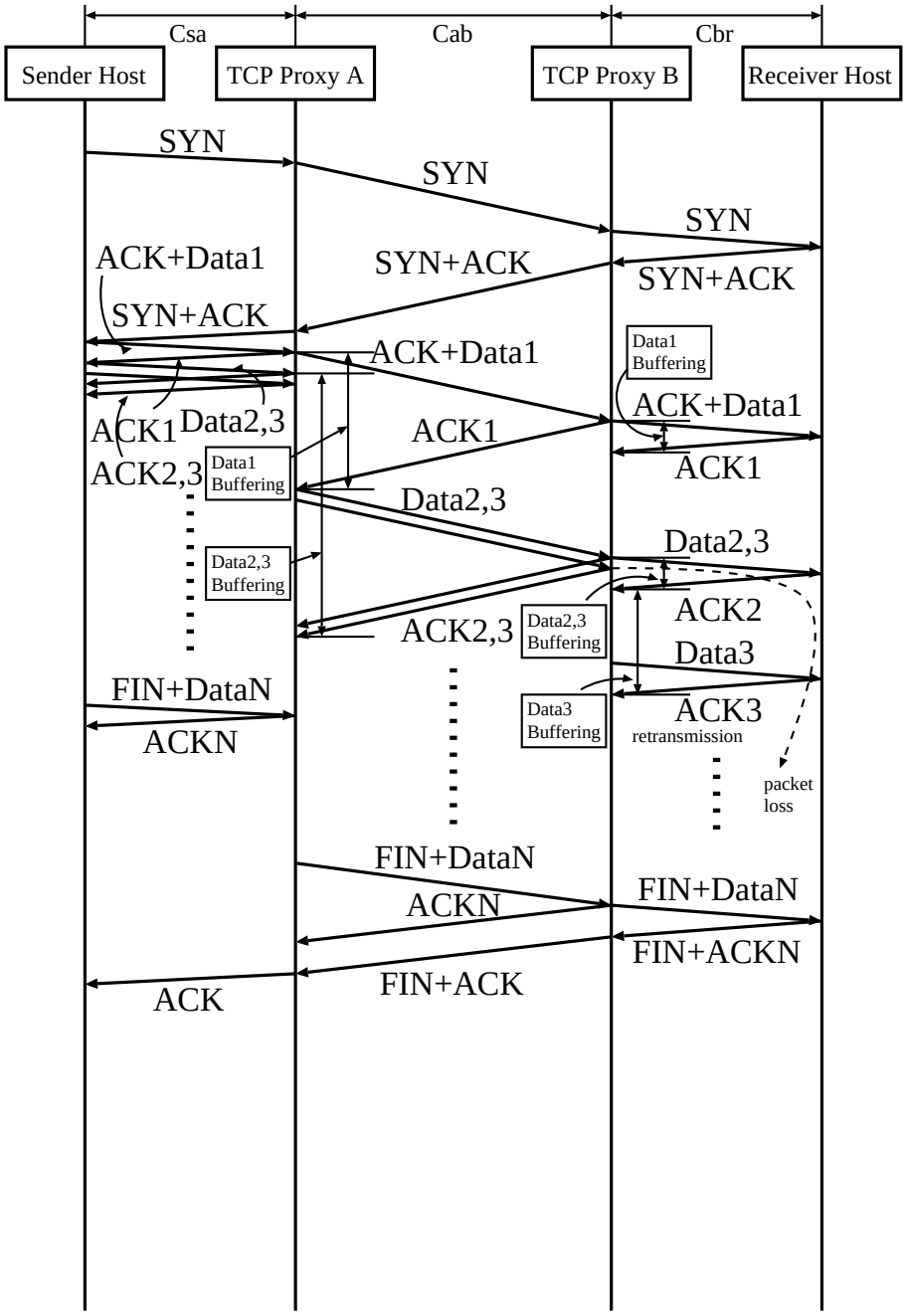


図 4: 擬似 ACK パケットを用いる TCP プロキシ機構におけるパケット転送

確実に転送されるため、ほとんどの場合においてはエンド端末間のパケット到達性は確保されると考えられる。また、コネクション設定・切断時に送信される SYN・FIN パケットに対しては、図 3 の場合と同様、擬似 ACK パケットを用いずに中継転送されるため、コネクション設定可能性は維持される。

このように TCP コネクションを分割しパケットを中継転送することによって、局所的なパケット再送や、擬似 ACK パケットの利用による性能向上だけでなく、様々な利点を得ることができる。分割された各 TCP コネクションは、それぞれが独立した送信側および受信側 TCP 制御を行うため、各分割コネクションが通過するネットワークの性質 (リンク帯域、RTT、パケット廃棄率、MTU (Maximum Transfer Unit: 最大転送パケットサイズ) など) に応じた調整を行うことができる。例えば、TCP プロキシ B と受信側端末間のネットワークが無線回線を含む場合には、TCP コネクションを分割するこによって、無線回線の高いパケット廃棄率、遅延の変動の影響を、他の分割 TCP コネクションが受けることを避けることができる。また、コネクション C_{br} に無線回線の特性を考慮した輻輳制御機構を持つ TCP [14, 15] を用いることで、さらにデータ転送速度が向上する。同様に、図 4 の TCP プロキシ A、B 間のネットワークが衛星回線を含む場合には、衛星回線の大きな帯域遅延積を考慮し、コネクション C_{ab} が用いる送受信ソケットバッファを大きくすることで、プロキシ A、B 間のデータ転送速度が向上し、全体のスループットが改善される。

また、TCP プロキシにおいて複数の TCP コネクションを収容し、分割中継を行うことで、ユーザ間の公平性の維持や、サービスの差別化を行うことも可能であると考えられる。例えば、TCP プロキシ A、B 間で HighSpeed TCP [16] などの特殊な機構によって高速データ転送を可能とする TCP を用いることで、サーバ間の大容量データ転送のみを高速化する、といった目的にも用いることができる。また、これらの制御を、送受信端末の TCP を改変することなく実現することが可能であることも、レイヤ 4 オーバレイネットワークアーキテクチャの大きな利点である。

3　実装

本章では、本報告において用いた TCP プロキシ機構の実装の詳細について述べる。本報告における実装では、中継処理を高速化するための様々な方式に関しては考慮しておらず、アプリケーション層において簡易的に TCP プロキシ機構を実装している。この実装を用いて処理遅延時間等の性能評価を行うことで、TCP プロキシ機構の基本的な性能に関する議論を行う。以降では、本報告で対象としている、PC ルータ上での実装およびネットワークプロセッサシステム上での実装について述べる。

3.1　PC ルータ上での実装

図 5 は、PC ルータ上に実装する TCP プロキシ機構の仕組み、およびパケット処理の様子を示している。PC ルータに到着したパケットは、ネットワークインタフェースカード (NIC) のバッファからダイレクトメモリアクセス (DMA) によってシステムカーネル内のメモリに移動される。その後、IP 処理および TCP 処理を受け、パケットのペイロード部分は受信ソケットバッファ(Socket Buffer) へ格納される。その後、TCP プロキシを実現するアプリケーションがデータをカーネルメモリからアプリケーションバッファへコピーする。その後、中継先の TCP コネクションの送信ソケットバッファへコピーされ、TCP/IP 処理を受けた後、パケットの転送が行われる。このように、TCP プロキシ機構は対応する中継コネクションの送受信ソケットバッファ間のデータの受け渡しを行うことにより、パケットデータの中継を行う。

この TCP プロキシ方式は、非常に単純な機構を持つが、アプリケーション層で中継処理を行うため、カーネルとアプリケーション間で 2 回のメモリコピーが発生する。したがって、大きな処理遅延時間が発生する可能性がある。この処理遅延を回避するための方式として、データ送受信時に発生するカーネルとアプリケーション間のメモリコピー処理を回避する、ゼロコピー方式 [17-19] に関する研究が行われている。また、ネットワークインターフェースカードのバッファからアプリケーションのメモリ空間へ直接データパケットを移動させる RDMA (Remote Direct Memory Access) [20] と呼ばれる方式が提案されている。これらの

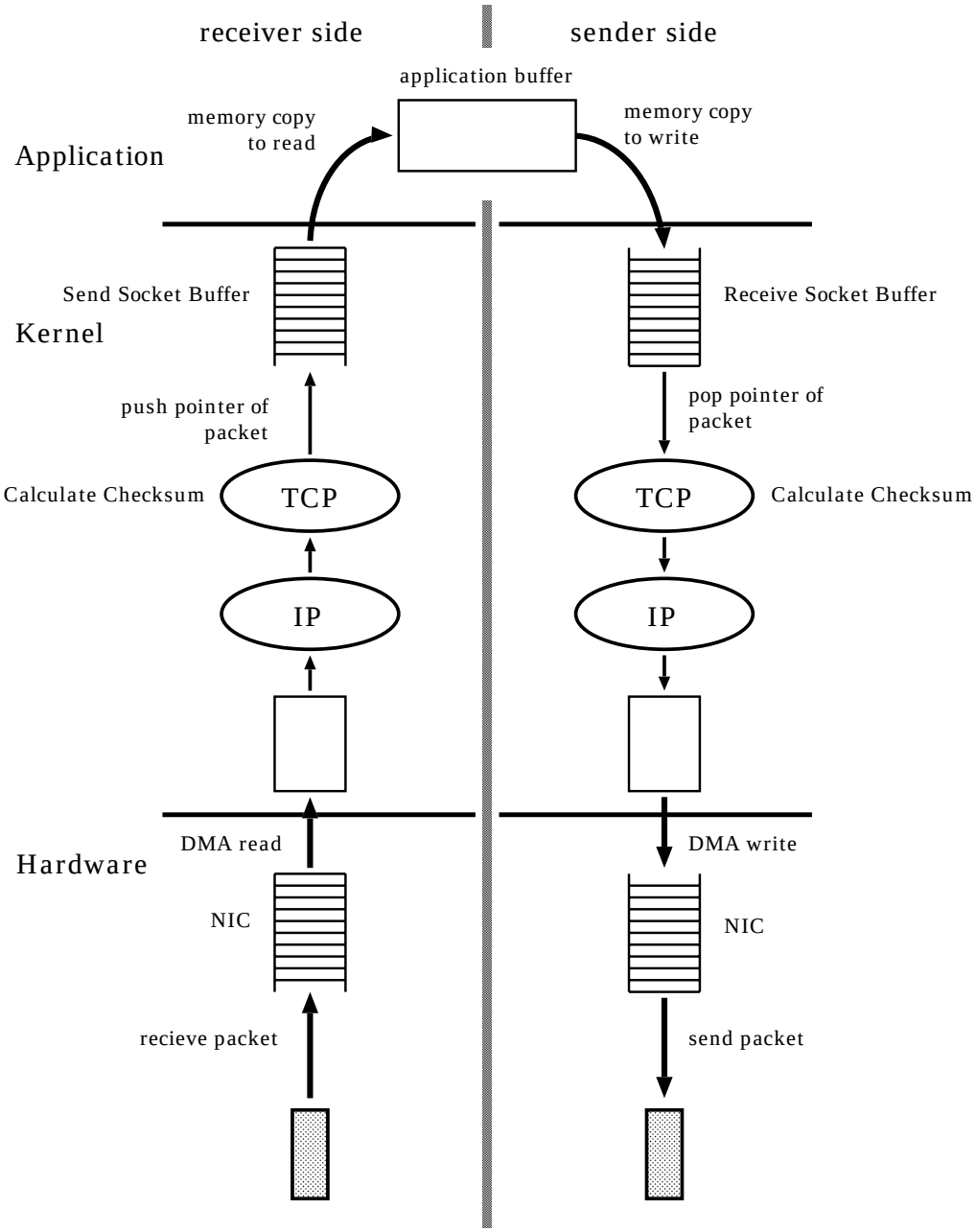


図 5: PC ルータ上での TCP プロキシ機構の実装概要

方式を用いることで、TCP プロキシ機構の処理速度を大幅に向上させることが可能である。本報告ではこれらの高速化手法の実装、性能評価は直接行なわないが、メモリコピー処理を含む簡易 TCP プロキシ機構の処理オーバーヘッドの評価を詳細にわたって行うことで、TCP プロキシ機構の実現に必要な処理にかかる時間を、実装を行うシステムの能力を用いて推測することが可能となるため、これらの手法を適用した場合の性能を実装・計測を行うことなく予測することができることを示す。

3.2 ネットワークプロセッサシステム上での実装

ネットワークプロセッサは、パケット処理に適したハードウェア構成、および命令セットを持つため、高速パケット処理が可能なネットワークノードを構築することができると考えられる。さらに、パケット処理部はプログラマブルあるため、機能拡張性に優れ、プログラム資源を再利用することが可能である。本報告では、ネットワークプロセッサシステムとして、IXP1200 [21] を搭載するネットワークプロセッサシステムの評価ボードである ENP2505 [22] を用いる。

図 6 は IXP1200 内における各ユニットの配置図を、図 7 は IXP1200 と周辺ハードウェアの配線図をそれぞれ示す。IXP1200 はパケット処理に関するプログラミングを行うことができるマイクロエンジンを 6 個、および主にマイクロエンジンの制御用途に用いられる StrongARM (SA) コアを 1 個の、合計 7 個の RISC プロセッサを持ち、外部メモリを扱うためのインタフェースである SRAM ユニット、および SDRAM ユニットを有する。さらに、パケット等のデータ転送用内部バスとして PCI ユニットや IX-Bus ユニットと呼ばれる独自のバス機構を有する。図 8 に示す ENP2505 は、IXP1200 チップを中心として、4 つのファストイーサネットのネットワークインタフェース、SRAM、および SDRAM 等から構成される、ネットワークプロセッサシステムボードである。本報告ではこの ENP2505 上に TCP プロキシ機構を実装する。

図 9 は、ENP2505 を用いた TCP プロキシ機構の仕組みを示した図である。ボード上のネットワークインターフェースに入ってきたパケットは、IXP1200 チップ内のマイクロエンジンによって入力処理が行われ、SA に引き渡される。TCP/IP の処理および TCP プロキ

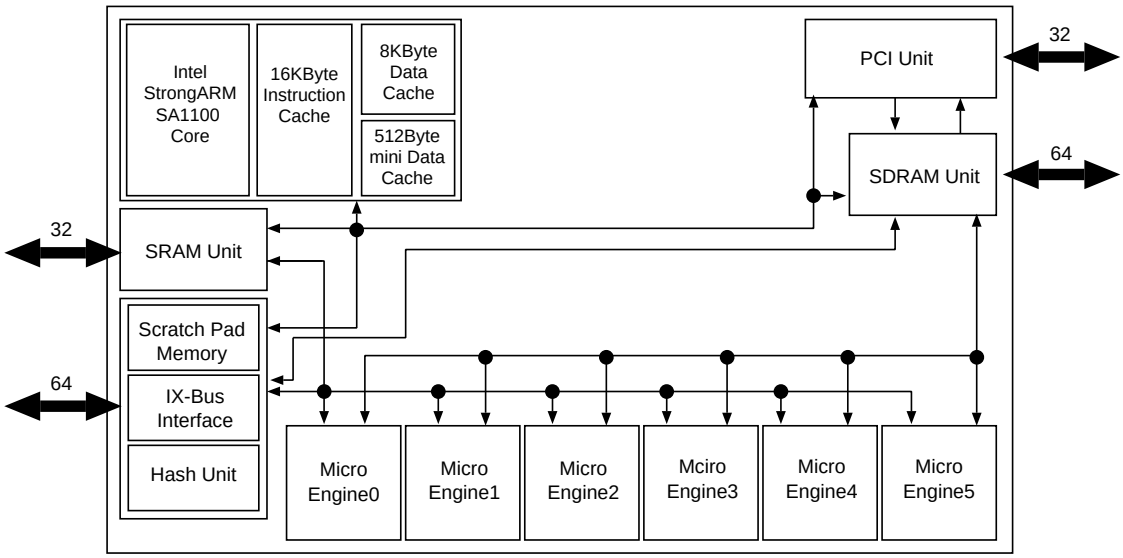


図 6: IXP1200 内のユニット配置

シ機構の処理は SA によって行われ、再度マイクロエンジンにパケットが引き渡される。その後、パケットはマイクロエンジンによる送出処理を受け、ネットワークインターフェースから出力される。

マイクロエンジンによるパケット入出力処理には、インテル社が提供しているパケット入出力のリファレンスデザインである SRD (Simplified Reference Design) [23] を改良したプログラム [24] を用いる。このプログラムは IP ルーティング処理を行う機能も持っているため、5 章における実験では、マイクロエンジンで IP 処理を行う場合の性能評価もあわせて行う。SA 上では BlueCat Linux [25] をベースとした Embedded Linux システムが動作し、TCP/IP 処理、および TCP プロキシ処理を行う。SA 上で動作るす TCP プロキシ処理に関しては、前節で示した PC ルータ上の実装と同じ方式を用いる。また、SA 上の Linux システムは ENP2505 上の SDRAM をメインメモリとして用いる。これらの機能分担は、TCP では複雑なプロトコル処理を行うため、マイクロエンジンで行うことができないこと、また SA 上で Linux が動作するため、Linux が持つ安定した TCP/IP 処理の実装を用いることができることを考慮して決定している。

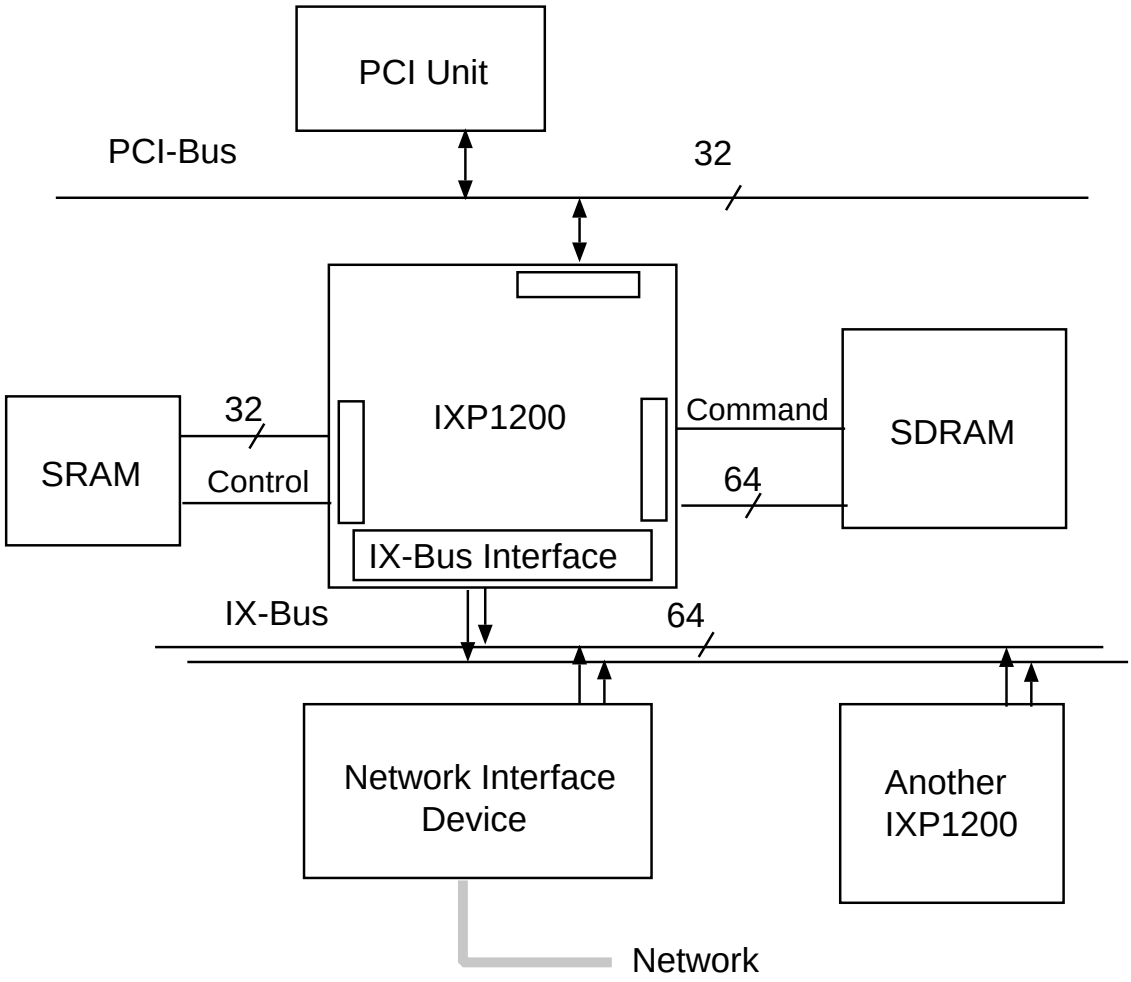


図 7: IXP1200 と周辺ハードウェアの配線図

図 8: ネットワークプロセッサボード ENP2505

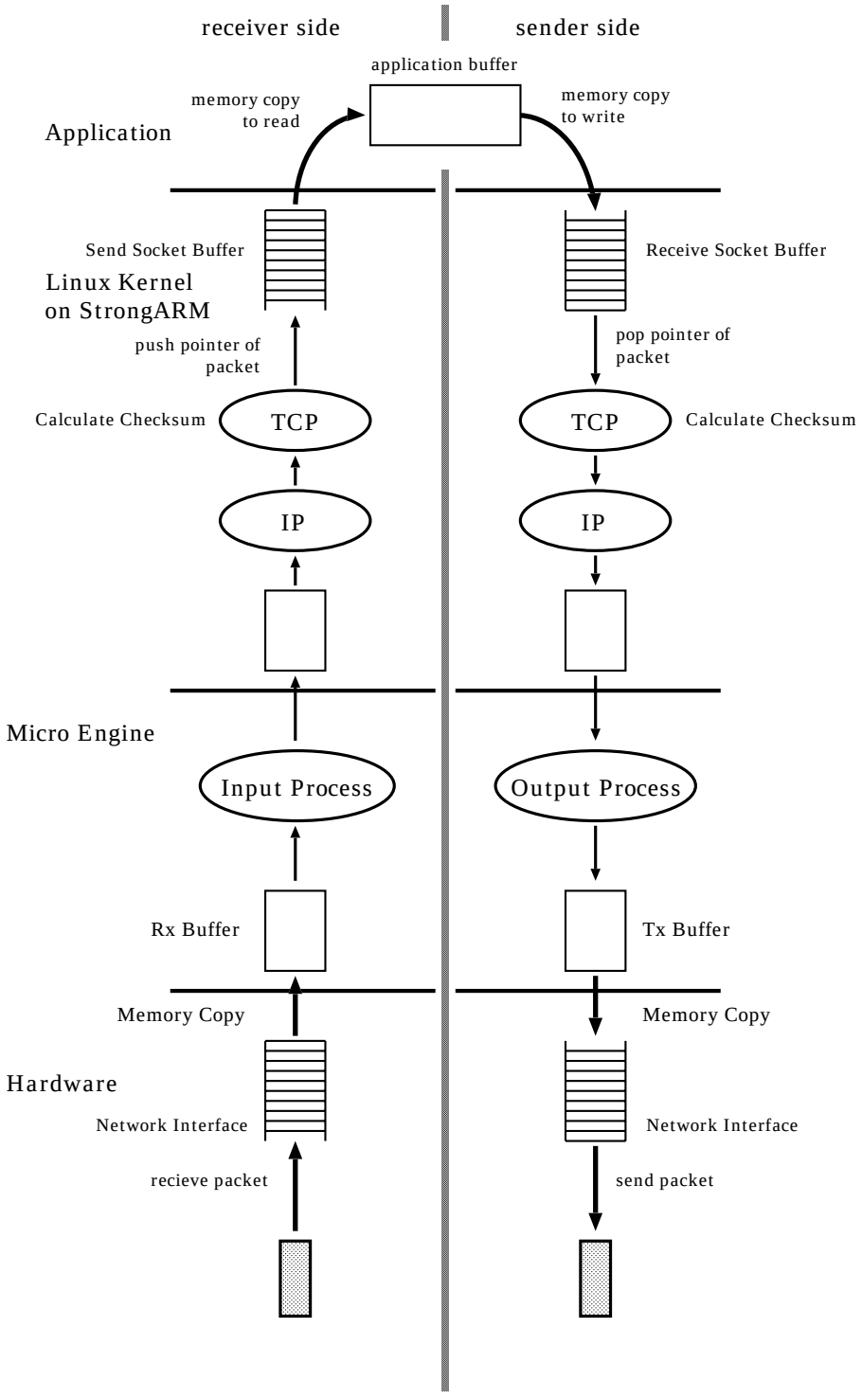


図 9: ネットワークプロセッサ上での TCP プロキシ機構の実装概要

4　TCP プロキシ機構の処理遅延時間の推定

本章では、3 章で示した TCP プロキシ機構の実装に基いて、実装を行う PC ルータ及びネットワークプロセッサシステムの仕様や能力を考慮した、1 パケットあたりの処理遅延時間の推測を行う。また、その結果を用いて TCP プロキシ機構の処理性能、およびそのボトルネックに関する議論を行う。本章において示した推測結果は、次章で示す実験結果との比較を行うことによりその妥当性を評価する。

4.1　PC ルータの場合

図 5 に示したように、PC ルータ上に実装された TCP プロキシ機構においては、到着したパケットは CPU による TCP/IP のプロトコル処理を受けた後にアプリケーションへ引き渡さる。その後中継処理を行い、中継先の TCP コネクションへパケットが引き渡され、TCP/IP の処理を受けてネットワークへ出力される。TCP/IP のプロトコル処理、およびカーネルとアプリケーション間のメモリコピー処理にかかる時間は、PC ルータの CPU 能力およびメモリアクセス能力に依存する。表 1 は、本報告において TCP プロキシ機構の実装を行った PC の CPU 能力およびメモリアクセス能力を示している。CPU 能力に関しては、物理的な動作周波数ではなく、Linux システムにおいて CPU 能力を示す目安である BogoMIPS 値を用いる。BogoMIPS 値は CPU の処理能力を比較評価するには適さない [26] が、ここでは簡単のために、CPU 処理能力の目安として用いる。メモリアクセス能力に関しては、メモリのハードウェア仕様からメモリアクセス速度を算出するのではなく、ベンチマークツール LMbench [27] を用いて計測したメモリアクセス速度を用いる。LMbench は、UNIX システムにおけるメモリキャッシュ能力、メモリアクセス速度、ディスクアクセス速度等を計測することができるベンチマークツールであり、メモリアクセス速度に関しては、64 MByte のデータのリード、ライト、コピー処理を行う時の速度を計測している。したがって、より現実的なメモリアクセス時間の推定が可能になると考えられる。

ネットワークインターフェースカードとシステムカーネル間のパケット転送は DMA (Direct Memory Access) 処理によって行われる。DMA 処理は高速であるため、ここでは遅延時間

表 1: 実験で用いた PC ルータの処理能力

CPU [BogoMIPS]	3381.65
Main Memory Size [MB]	256
Memory Copy Speed [MB/sec]	719.2
Memory Read Speed [MB/sec]	1407
Memory Write Speed [MB/sec]	1022

表 2: PC ルータ上の実装における TCP プロキシ機構の処理時間

処理内容	導出過程	MTU [Bytes]						
		250	500	1000	1500	3000	6000	9000
IP input (data packet)	57 $[ins]$/3381.65 M	16	16	16	16	16	16	16
TCP input (data packet)	187 $[ins]$/3381.65 M	55	55	55	55	55	55	55
Memory Read (for checksum)	$Size$ $[B]$/1407 $[MB/sec]$	177	355	710	1066	2132	4264	6396
Memory Copy (mbuf to buffer)	$Size$ $[B]$/719.2 $[MB/sec]$	347	695	1390	2085	4171	8342	12513
TCP output (ack packet)	235 $[ins]$/3381.65 M	69	69	69	69	69	69	69
IP output (ack packet)	61 $[ins]$/3381.65 M	18	18	18	18	18	18	18
Memory Copy (buffer to mbuf)	$Size$ $[B]$/719.2 $[MB/sec]$	347	695	1390	2085	4171	8342	12513
TCP output (data packet)	235 $[ins]$/3381.65 M	69	69	69	69	69	69	69
Memory Read (for checksum)	$Size$ $[B]$/1407 $[MB/sec]$	177	355	710	1066	2132	4264	6396
IP output (data packet)	61 $[ins]$/3381.65 M	18	18	18	18	18	18	18
合計 [ns]		1293	2345	4445	6547	12851	25457	38063

の算出に際しては考慮しない。パケットが受ける TCP/IP のプロトコル処理に要する時間に関しては、文献 [28] に示されている TCP/IP プロトコル処理の命令数を用いる。文献 [28] では BSD システムを対象とした命令数を導出しているが、Linux システムにおいてもほぼ変らない命令数である。メモリアクセス処理は、パケット送受信時のチェックサム値の計算、および TCP プロキシ機構を実現するアプリケーションとカーネル間のデータ移動の際に発生する。ここでは簡単のため、メモリアクセス処理にかかる時間は、処理データサイズに比例すると仮定する。

表 2 は、図 5 に示した TCP プロキシ機能の処理内容および表 1 の処理能力に基づいて算出した、1 パケットあたりの TCP プロキシ処理遅延時間の推定結果である。パケットサイズは 250–9000 Bytes としている。この結果から、全体の処理遅延のほとんどを、メモリアクセス処理にかかる時間が占めていることがわかる。すなわち、TCP/IP のプロトコル処理にかかる時間は、データの移動、チェックサム等メモリアクセスをともなう処理のそれに比べて非常に小さいといえる。これは、文献 [20, 28, 29] などにおいて得られている、TCP を用いたデータ送受信における送受信処理遅延時間の解析結果と同様である。近年のコンピュータハードウェア技術の向上によって、CPU の処理速度は大幅に向上したが、メモリアクセス速度の向上の度合いはそれに比べて小さいため [30]、メモリアクセス処理がシステム全体の処理能力の大きなオーバヘッドになるという傾向は今後より大きくなると考えられる。

4.2 ネットワークプロセッサシステムの場合

本節では、ENP2505 上における TCP プロキシ機構の実装に関して、前節と同様の手法を用いた処理遅延時間の推定を行う。表 3 は、SA の CPU 処理能力とメモリアクセス処理能力を示している。CPU 処理能力に関しては PC ルータの場合と同様に BogoMIPS 値を用いている。また、PC ルータにおいて用いたベンチマークツールである LMbench は SA 上の Linux システム上では動作しないため、LMbench のうち、メモリアクセス処理に関するベンチマーク部分だけを抜粋したプログラムを用いて、メモリ処理能力の計測を行った。なお、LMbench からメモリアクセス処理のベンチマーク部分だけを抜粋したプログラムの妥当性は、PC ルータにおけるベンチマーク結果を比較することによって確認した。この結果

から、ENP2505 のメモリアクセス速度が PC ルータに比べて非常に小さいことがわかる。これは、SA 上の Linux システムは ENP2505 が持つ SDRAM をメインメモリとして用いるため、Linux カーネルやアプリケーションがメモリアクセスを行うためには、IXP1200 内のマイクロエンジンが SDRAM へのアクセスを行う際と同様の処理が必要となるためである。次に表 4 に、図 9 に示した実装概要と、表 3 に示した処理能力を基に、ENP2505 上における TCP プロキシ機構の 1 パケットあたりの処理遅延時間を示している。

この表から、PC ルータ上における実装と同様、処理遅延時間の大部分をメモリアクセス処理が占めていることがわかる。これは、ENP2505 の SA は動作周波数が 200MHz と低速であるが、上述のようにメモリアクセス速度が非常に低速であるため、PC ルータ上の実装と同様、メモリアクセス能力が性能のボトルネックとなっていることが原因である。また、PC ルータ上の実装 (表 2) と比較すると、1 パケットあたりの処理遅延時間が、MTU の大きさにかかわらず約 20 倍になっていることがわかる。このことから、ENP2505 上の TCP プロキシ機構の処理能力が PC ルータ上の実装に比べて約 20 分の 1 になると推測される。これらの推測結果の妥当性は、次章において実験結果との比較を行うことによって評価する。

表 3: ENP2505 の処理能力

CPU [BogoMIPS]	217.91
Main Memory Size [MB]	62
Memory Copy Speed [MB/sec]	31.06
Memory Read Speed [MB/sec]	80.95
Memory Write Speed [MB/sec]	64.23

表 4: ENP2505 上での TCP プロキシ機構の実装における処理時間

処理内容	導出過程	MTU [Bytes]			
		250	500	1000	1500
IP input (data packet)	57 $[ins]$/217.91 M	261	261	261	261
TCP input (data packet)	187 $[ins]$/217.91 M	858	858	858	858
Memory Read (for checksum)	$Size$ $[B]$/80.95 $[MB/sec]$	3088	6176	12353	18529
Memory Copy (mbuf to buffer)	$Size$ $[B]$/31.06 $[MB/sec]$	8048	16097	32195	48293
TCP output (ack packet)	235 $[ins]$/217.91 M	1078	1078	1078	1078
IP output (ack packet)	61 $[ins]$/217.91 M	279	279	279	279
Memory Copy (buffer to mbuf)	$Size$ $[B]$/31.06 $[MB/sec]$	8048	16097	32195	48293
TCP output (data packet)	235 $[ins]$/217.91 M	1078	1078	1078	1078
Memory Read (for checksum)	$Size$ $[B]$/80.95 $[MB/sec]$	3088	6176	12353	18529
IP output (data packet)	61 $[ins]$/217.91 M	279	279	279	279
合計		26105	48379	92929	137477

5　TCP プロキシ機構の処理能力の計測

本章では、TCP プロキシ機構を実装した PC ルータおよびネットワークプロセッサシステムを用いてデータ転送実験を行い、4 章で行った処理遅延時間の推測結果の妥当性を評価する。また、それらの結果を基に TCP プロキシ機構の処理性能を向上させるための指針を示し、高速化手法を適用したり、将来の高速なネットワークプロセッサシステムを用いた場合の性能予測を行う。

図 10 に、本章での実験環境を示す。実験環境は、送信側端末、受信側端末、およびそれらを接続する中継ノードから構成される。送受信端末は共に Pentium4 1.70 GHz の CPU、256 MB の RDRAM メモリを持つ PC である。中継ノードにはイーサネットのスイッチングハブ、送受信端末と同じ PC、およびネットワークプロセッサシステム ENP2505 を用いる。送受信端末とノードの接続には、ファストイーサネットおよびギガビットイーサネットを用いる。実験においては、中継ノードをイーサネットのスイッチングハブ、IP ルータ、TCP プロキシノードとして用い、送信側端末から受信側端末に向けてデータ転送を行った際のデータ転送スループットを計測する。中継ノードが TCP プロキシとして動作する場合には、送受信端末間の TCP コネクションを 2 つのコネクションに分割し、パケットの中継転送を行う。またその場合には、TCP プロキシノードにおける処理遅延時間の計測を行う。

送受信端末および PC ルータは RedHat Linux 7.2 [31] システムを用いる。カーネルのバージョンは 2.4.18 である。また、実験において設定される TCP コネクションの送受信バッファサイズ、および送信側 TCP の最大ウィンドウサイズは十分大きいとする。送受信端末とノード間の伝搬遅延時間はともに約 5 msec である。

5.1　PC ルータの場合

最初に、中継ノードに PC を用いる場合の結果を示す。図 11 は、ネットワークにファストイーサネットを用いてデータ転送を行った時の、ネットワークの MTU とスループットの関係を示している。図中には、中継ノードを IP ルータとして用いた場合 (PC-L3)、TCP プロキシとして用いた場合 (TCPproxy) に加え、中継ノードにファストイーサネットのスイッチ

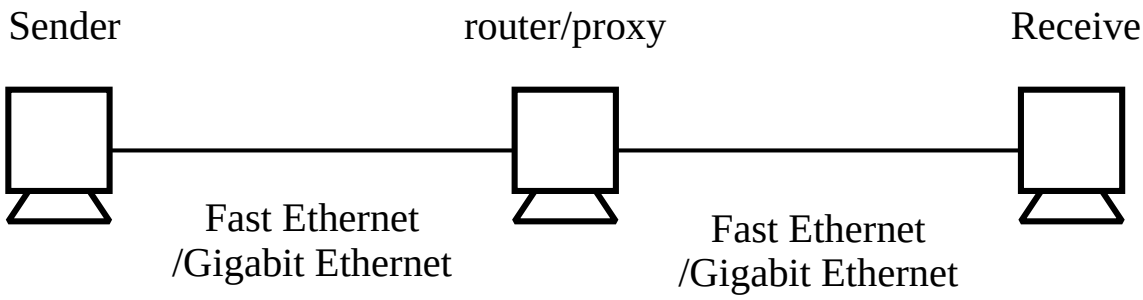


図 10: 実験ネットワーク環境

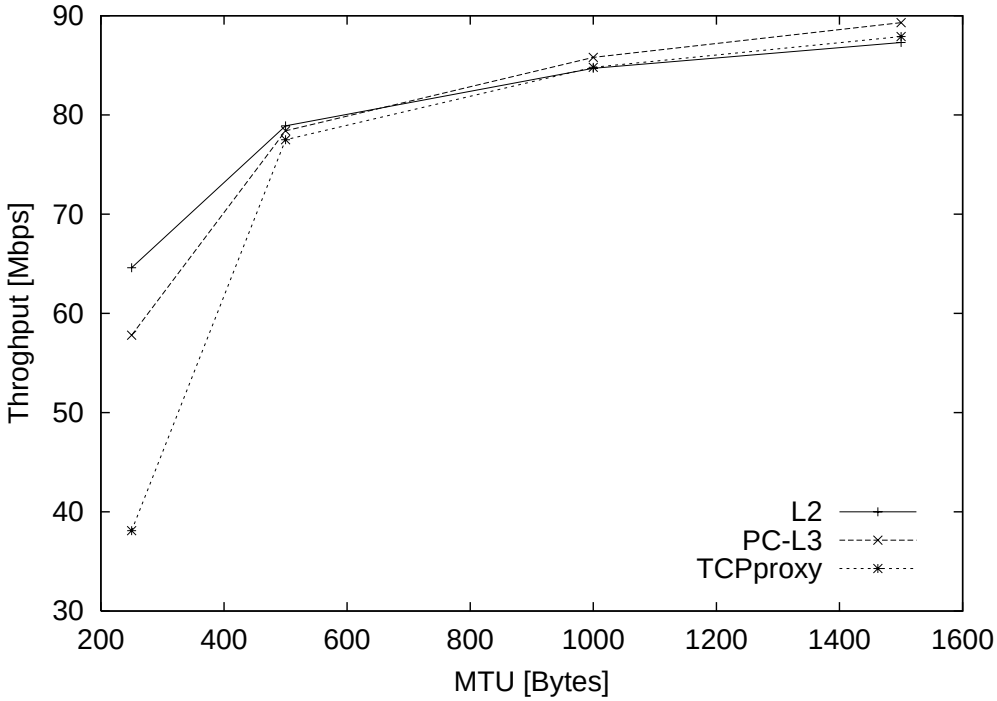
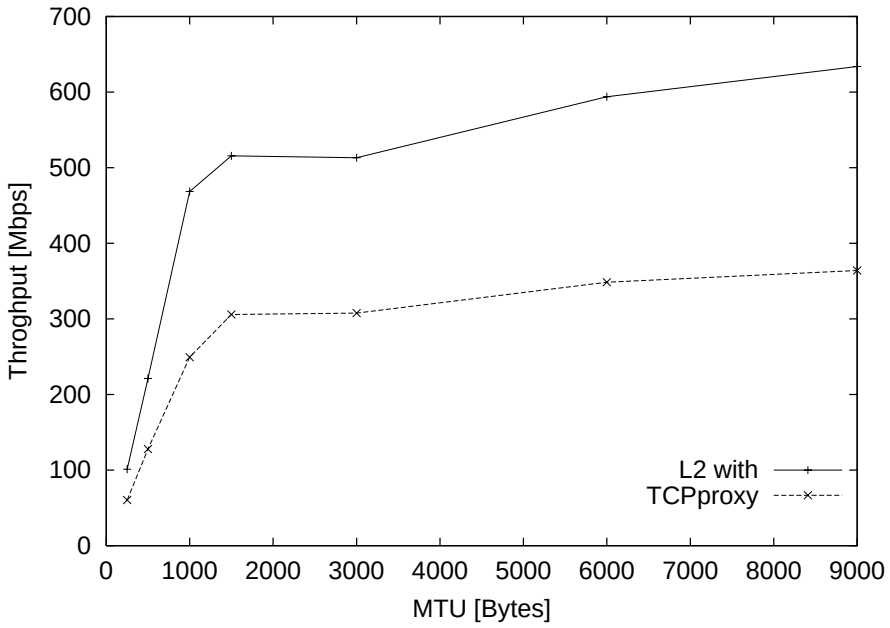


図 11: ファストイーサネット環境における MTU と処理性能の関係

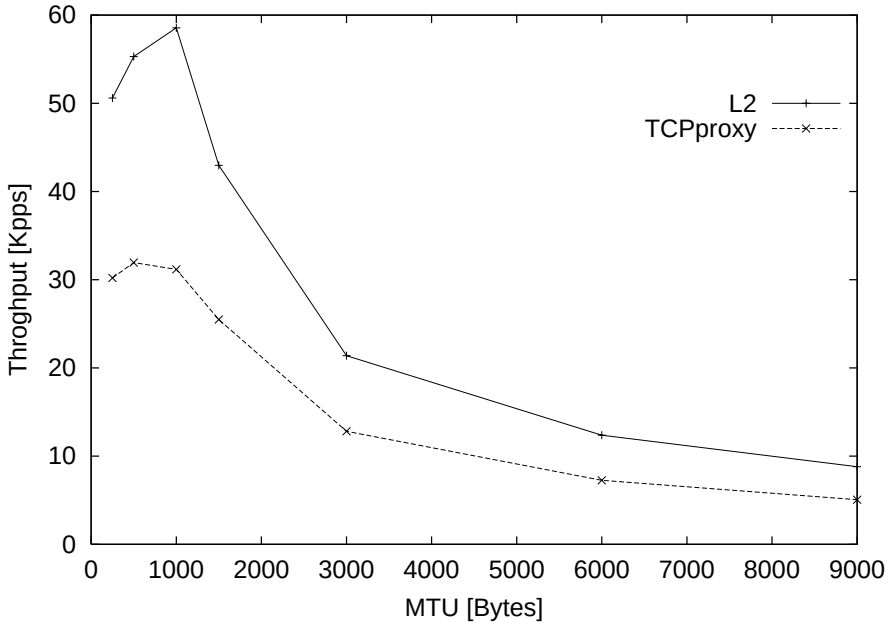
ングハブを用いた場合 (L2) の結果を示している。図から、MTUが大きい場合には、L2 および PC-L3 においてはファストイーサネットの限界にほぼ近いスループット (約 90 Mbps) が得られていることがわかる。また、MTUが小さくなるとスループットが低下しているが、これはパケットに含まれるヘッダの割合が大きくなるためであると考えられる。また、TCP プロキシ機構は、MTUが大きい場合には、L2 および PC-L3 と同等のスループットが得られているが、MTUが 250 Bytes の場合には、L2 および PC-L3 (約 60 Mbps) に比べて低下していることがわかる。これは、MTUが 250 Bytes の場合、60 Mbps の処理速度は 1 パケットあたり 330 nsec の処理時間に相当し、4 章における推定結果よりも小さな処理時間になっていることからも明らかである。

次に、ネットワークにギガビットイーサネットを用いた場合の実験結果を図 12 に示す。図には、中継ノードをイーサネットのスイッチングハブ (L2)、および TCP プロキシ (TCPproxy) として動作させた場合の実験に関して、スループットとして 1 秒あたりの処理ビット数 (図 12(a)) およびパケット処理能力として 1 秒あたりの処理パケット数 (図 12(b)) を示している。図 12(a) から、MTU の値にかかわらず、TCPproxy の性能が L2 の性能 (最大約 600 Mbps) に比べて大幅に低く、約 350 Mbps にとどまっていることがわかる。これは、ギガビットイーサネットによって高速化されたネットワークの処理速度に、TCP プロキシ機構の処理速度が追随していないことを示している。MTUが 9000 Bytes の場合、700 Mbps の処理速度は 1 パケットあたり 103 nsec の処理時間に相当し、4 章における推定結果よりも小さな処理時間になっていることからも明らかである。このことは、図 12(b) に示す、パケット処理能力の変化を見ても明らかである。図から、MTUが大きくなるにつれて、TCPproxy の最大パケット処理速度が約 30 Kpps を上限に小さくなっていることがわかる。これは、MTUが大きくなるとメモリコピー処理により大きな処理時間を必要とするため、1 パケットあたりの処理遅延時間が大きくなるためである。またこの結果により、メモリコピー処理能力が TCP プロキシ機構の処理性能の最大のボトルネックになっているという、4 章における処理遅延時間の推測結果の妥当性が示されたといえる。

次に、TCP プロキシ処理を行われた各パケットについて、中継処理にかかった時間を計測した結果を示す。ここで中継処理とは、中継元の TCP コネクションの受信ソケットバッ



(a) スループット (1 秒あたりの処理ビット数)



(b) パケット処理能力 (1 秒あたりの処理パケット数)

図 12: ギガビットイーサネット環境における MTU と処理性能の関係

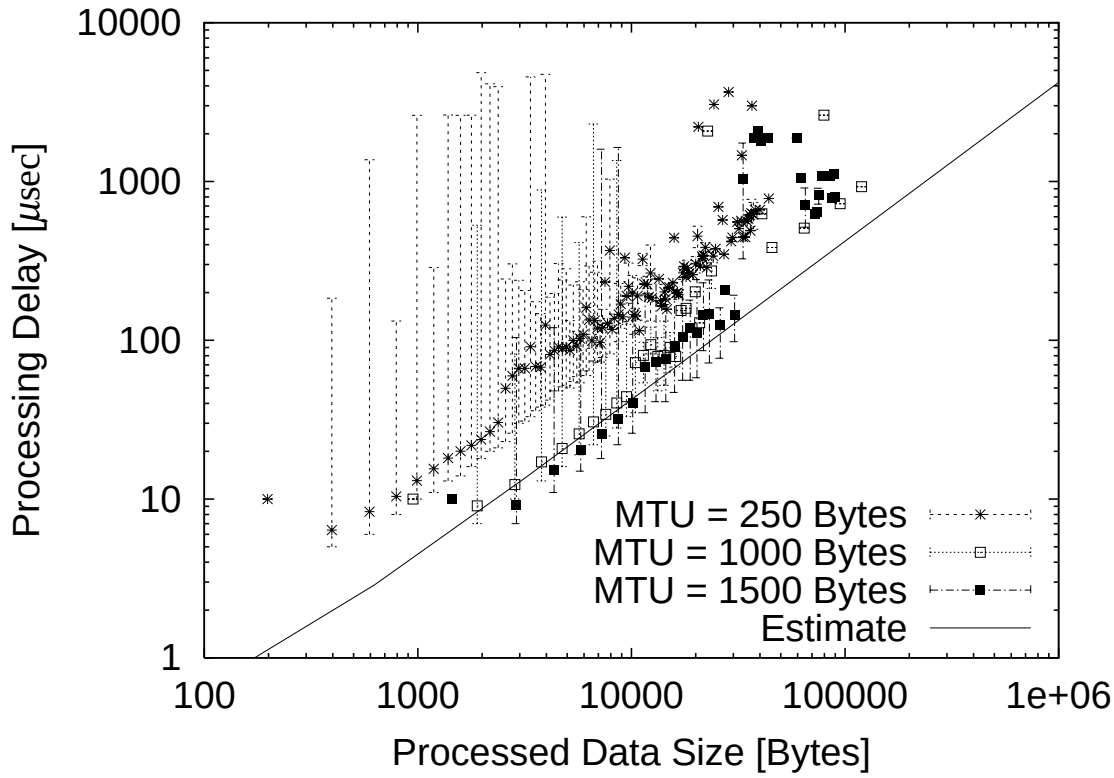


図 13: 処理データサイズと処理時間の関係

ファからパケットをコピーし、中継処理を行い、中継先の TCP コネクションの送信ソケットバッファへパケットを書き込むまでの処理を意味する。また、本報告で実装した TCP プロキシ機構のアプリケーションは、中継元の TCP コネクションの受信ソケットバッファに書き込まれたデータを 1 パケットずつ処理するのではなく、処理を開始した時点で受信ソケットバッファに格納されている全てのデータを一度に処理する。図 13 は、一度に処理を行うデータサイズと処理時間の平均値、最小値および最大値の関係を示している。図には、4 章における推測結果 (Estimate) をあわせて示している。

図から、一度に処理を行うデータサイズが大きい場合には、処理遅延時間の平均値が 4 章における推測結果とよく適合しており、処理遅延時間の推測手法が妥当であることを示している。しかし、データサイズが小さい場合には、処理遅延時間が推測結果より大きくなっている。これは、処理するデータサイズが小さい場合には、そのデータを格納するためのメモ

リ空間や、コピー処理にオーバーヘッドが発生するためであると考えられる。また、4 章において示した LMbench を用いたベンチマークが、64MB のデータに対するメモリアクセスの性能を計測していることも、要因の一つであると考えられる。

また、同じデータサイズの場合においも、処理遅延時間に大きなばらつきがあることもわかる。これは、中継処理そのものは十分高速に行われるが、中継先の TCP コネクションからのパケット転送がそれに比べて低速であるために、アプリケーションから中継先の TCP コネクションの送信ソケットバッファへのメモリコピー処理が終了しないことが原因である。これにより、次の受信データの処理にも遅延が発生し、4 章における推測結果よりも全体の処理遅延時間が大きくなっている。

これらの結果から、4 章で示した処理遅延時間の推測方式は、ネットワークへのパケット送信がボトルネックになり遅延が増大する場合を除くと、ほぼ正確に TCP プロキシ機構の処理遅延時間を推定していることがわかる。したがって、TCP プロキシ機構を実装するノードの能力が変化したり、ゼロコピー手法などの高速化手法を導入した場合における、TCP プロキシ機構の処理性能をある程度予測できると考えられる。例えば、前節で用いた PC ルータ上の TCP プロキシ機構にゼロコピー手法を導入し、カーネルと中継アプリケーション間のコピー処理が省略された場合の処理遅延時間の推定結果を表 5 に示す。この場合、表 5 における 2 度のメモリコピー処理にかかる遅延時間が省略される、全体の処理遅延時間が約 1/2–1/3 に短縮されている。したがって、ゼロコピー手法を導入することによって、TCP プロキシ処理性能が約 2–3 倍になると考えられる。

5.2 ネットワークプロセッサシステムの場合

本節では、中継ノードにネットワークプロセッサシステム ENP2505 を用いた場合の実験結果を示す。図 14 は、転送時に用いた MTU とデータ転送スループットの関係を示している。ここでは、イーサネットのスイッチングハブを用いた場合 (L2)、ENP2505 のマイクロエンジンで IP ルーティング処理を行った場合 (NWP-L3)、および ENP2505 を TCP プロキシとして用いた場合 (NWP-L4) の結果を示している。図では、スループット (1 秒あたりの処理ビット数) (図 14(a))、およびパケット処理速度 (1 秒あたりの処理パケット数) (図 14(b))

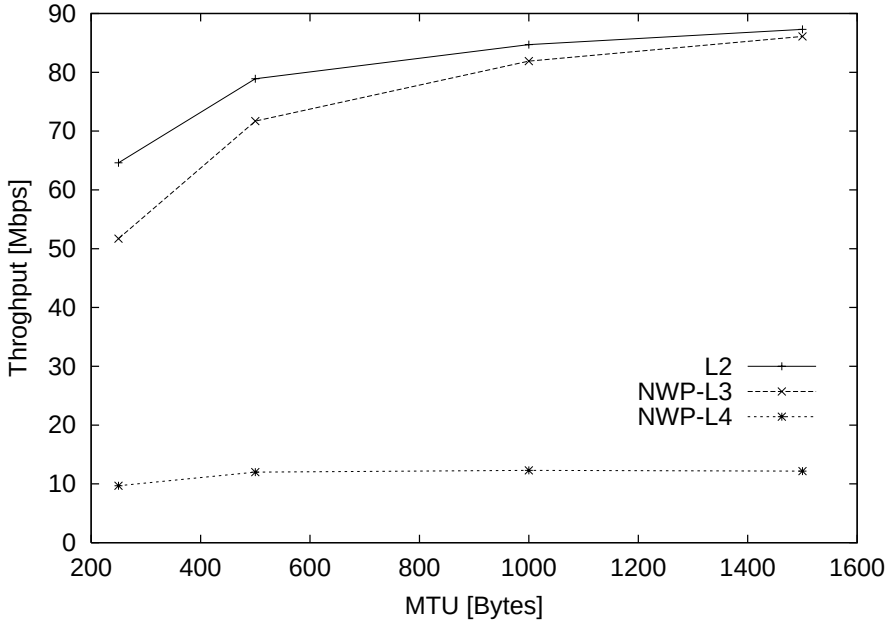
表 5: ゼロコピー手法を用いた場合の PC ルータ上の実装における TCP プロキシ機構の処理時間

処理内容	導出過程	MTU [Bytes]						
		250	500	1000	1500	3000	6000	9000
IP input (data packet)	57 $[ins]$/1700 M	16	16	16	16	16	16	16
TCP input (data packet)	187 $[ins]$/1700 M	55	55	55	55	55	55	55
Memory Read (for checksum)	$Size$ $[B]$/1407 $[MB/sec]$	177	355	710	1066	2132	4264	6396
TCP output (ack packet)	235 $[ins]$/1700 M	69	69	69	69	69	69	69
IP output (ack packet)	61 $[ins]$/1700 M	18	18	18	18	18	18	18
TCP output (data packet)	235 $[ins]$/1700 M	69	69	69	69	69	69	69
Memory Read (for checksum)	$Size$ $[B]$/1407 $[MB/sec]$	177	355	710	1066	2132	4264	6396
IP output (data packet)	61 $[ins]$/1700 M	18	18	18	18	18	18	18
合計 [ns]		599	955	1665	2377	4509	8773	13037

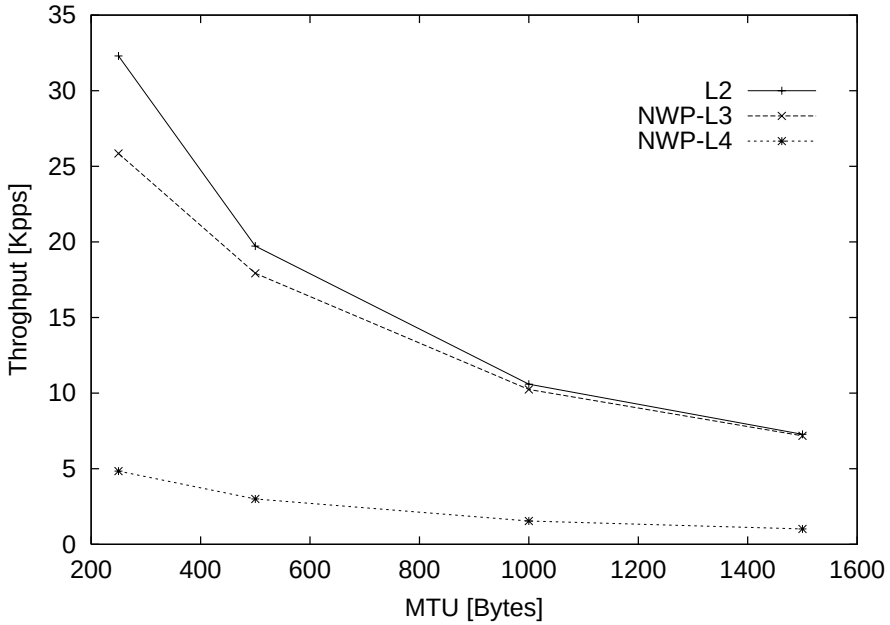
を示している。

図 14(a) から、ENP2505 のマイクロエンジンで IP ルーティング処理を行う場合 (NWP-L3) は、MTU の大きさにかかわらず、スイッチングハブを用いた場合 (L2) とほぼ同様のスループットを得ていることがわかる。これは、[24] において用いられているマイクロエンジンによる IP ルーティング処理が非常に高速に動作していることを示している。ところが、ENP2505 において TCP プロキシ処理を行う場合 (NWP-L4) には、スループットが大幅に低下している。これは、ENP2505 上に実装した TCP プロキシ機構は、SA にパケットを引き渡して処理を行うため、SA から SDRAM へのメモリアクセスが発生することが原因である。すなわち、4 章において示した推測結果 (表 4) からわかるように、SA 上の Linux のメモリアクセス速度が非常に低速であるために、全体のスループットが低下していると考えられる。

また、図 14(b) から、マイクロエンジンによる簡易的な IP ルーティング処理を行う場合 (NWP-L3) には、MTU が大きくなってもパケット処理速度は低下していないが、SA において TCP プロキシ処理を行う場合 (NWP-L4) は、パケット処理速度が大きく低下していることがわかる。これは、マイクロエンジンによるパケット処理速度は、ファストイーサネッ



(a) スループット (1 秒あたりの処理ビット数)



(b) パケット処理能力 (1 秒あたりの処理パケット数)

図 14: ネットワークプロセッサ上の実装における MTU と処理性能の関係

表 6: IXP2400 システムにおける TCP プロキシ機構の処理遅延時間

処理内容	導出過程	MTU [Bytes]			
		250	500	1000	1500
IP input (data packet)	57 $[ins]$/5400 M	10	10	10	10
TCP input (data packet)	187 $[ins]$/5400 M	34	34	34	34
Memory Read (for checksum)	$Size$ $[B]$/2428.5 $[MB/sec]$	102	205	411	617
Memory Copy (mbuf to buffer)	$Size$ $[B]$/931.8 $[MB/sec]$	268	536	1073	1609
TCP output (ack packet)	235 $[ins]$/5400 M	43	43	43	43
IP output (ack packet)	61 $[ins]$/5400 M	11	11	11	11
Memory Copy (buffer to mbuf)	$Size$ $[B]$/931.8 $[MB/sec]$	268	536	1073	1609
TCP output (data packet)	235 $[ins]$/5400 M	43	43	43	43
Memory Read (for checksum)	$Size$ $[B]$/2428.5 $[MB/sec]$	102	205	411	617
IP output (data packet)	61 $[ins]$/5400 M	11	11	11	11
合計		892	1634	3120	4604

トのネットワーク速度に比べて十分大きいため、パケットサイズに関係なく高速に処理を行うことができるが、SA において TCP プロキシ処理を行う場合は、メモリアクセス速度が性能のボトルネックとなっているため、表 4 に示したようにパケットサイズが大きくなるにつれて 1 パケットあたりの処理遅延時間が急激に増大するためである。

さらに、TCP プロキシ機構を PC ルータ上に実装した場合とスループットを比較すると、MTU の大きさが同じ場合には、中継処理のスループットが約 20 倍になっていることがわかる。これは、4 章における処理遅延時間の推定結果の比とほぼ一致している。これにより、4 章において示した処理遅延時間の推定結果が妥当であるといえる。したがって、本報告において示した TCP プロキシ機構の処理遅延時間の推測手法を用いることで、ネットワークプロセッサシステムの性能が変化した場合の TCP プロキシ処理性能を予測することが可能であると考えられる。たとえば本報告で用いている IXP1200 の後継にあたるネットワークプロセッサ IXP2400 [32] は、メモリが SDRAM から DDR-SDRAM に変更され、アクセス速度が約 30 倍に向上すると考えられる。さらに RISC プロセッサが 5400 MIPS で動作する XScale に変更されている [32]。これに基づいて、IXP2400 を用いた場合の処理性能を推測した結果を表 6 に示す。

この結果から、全体の処理遅延時間が約 1/30 になっているため、TCP プロキシ機構の処理性能が 30 倍に向上すると考えられ、ある程度高速なネットワークにおいても、ネットワークプロセッサを用いた TCP プロキシ機構を運用することが可能になると考えられる。

IXP1200 を用いたネットワークプロセッサシステム上における TCP プロキシ機構の高速化方式の 1 つとして、SA ではなくマイクロエンジンで TCP のプロトコル処理を行うことにより、SA とメモリとのアクセスを省略することが考えられる。しかし、IP のルーティング処理をマイクロエンジン上で行うシステムは [33] 等があるが、TCP の処理は非常に複雑であるため、マイクロエンジン上で実装することは困難であると考えられる。さらに、IXP1200 で用いられる StrongARM や IXP2400 上の XScale プロセッサ上では Linux 等の汎用のオペレーティングシステムが動作するため、これまでに十分検証が行われ、その動作が安定している TCP/IP の実装を用いることができる。したがって、将来のネットワークプロセッサシステム上における TCP プロキシ機構においても、TCP/IP 処理は StrongARM や XScale 上で行われることが予測される。すなわち、TCP プロキシ機構をネットワークプロセッサシステム上に実装した場合の処理性能を向上させるためには、StrongARM や XScale プロセッサが用いるメモリシステムへのアクセス速度を高速化させることが重要であるといえる。

また、TCP プロキシ機構の高速化手法として、TCP 処理をハードウェアで行う TCP オフロードエンジン (TOE: TCP Offload Engine) [34] を用いることが考えられる。しかし、現状で入手可能な TOE は、サーバ間の大容量高速転送のために使われることが多いため、その TCP は非常に簡略化されたものになっている。レイヤ 4 オーバレイネットワークにおいて行われる TCP プロキシ機構は、一般的なネットワークで用いられることが考えられるため、TOE を TCP プロキシ機構のために用いる場合には、既存の OS 上の TCP/IP の実装を反映した TOE を用いる必要がある。

6　おわりに

本報告では、レイヤ 4 オーバレイネットワークの実現に必要不可欠である、TCP プロキシ機構をネットワークプロセッサシステム ENP2505 上に実装し、その性能および処理オーバヘッドの評価を行った。評価の際には、PC ルータ上に実装した場合との比較を行い、ネットワークプロセッサを用いて実装することの有用性や特徴を評価した。その結果、現在のシステムにおいては、TCP プロキシ機構の処理性能のボトルネックになるのは、PC ルータ上の実装と同様、メモリアクセス処理であることが明らかとなった。またそれは、TCP/IP 処理および TCP プロキシ処理を行う ENP2505 上の StrongARM プロセッサと、パケットが格納されるメモリとのインタフェース機構が低速であることが原因であるため、TCP プロキシ機構の処理性能を向上させるためには、そのインタフェース速度を向上させることが必要であることを示した。

また、ネットワークプロセッサシステムおよび PC ルータ上での TCP プロキシ機構の実装における、1 パケットあたりの中継処理にかかる時間を、それぞれの CPU 処理能力、メモリアクセス能力等から簡単に推測する手法を示した。また、推測結果と実験結果を比較することによって、その推測結果が妥当であることを示し、それを基に、ゼロコピー方式などの高速化手法を適用した場合や、将来の高速なネットワークプロセッサシステム上での TCP プロキシ機構の処理性能を予測することができることを明らかにした。

今後の課題としては、ゼロコピー方式などを実装して TCP プロキシ機構を高速化し、本報告で行った性能予測が妥当であるかどうかの検討を行いたい。また、IXP2400 などを用いたより高速なネットワークプロセッサシステムを使った評価や、TCP プロキシ機構の処理性能として、スループットや処理遅延だけでなく、収容可能な TCP コネクション数に着目した評価をあわせて行う予定である。

謝辞

本報告を終えるにあたり、御指導、御教授を頂いた村田正幸教授に心より感謝致します。また、本報告において日頃から熱心に御指導を頂いた長谷川剛助教授に深く感謝致します。並びに、適切な助言を頂いた宮原秀夫教授、大阪府立看護大学医療技術短期大学の菅野正嗣助教授、サイバーメディアセンターの馬場健一助教授、宮原研究室の若宮直紀助教授、今瀬研究室の大崎博之助教授、大阪市立大学の阿多信吾助手、大阪大学大学院経済学研究科の荒川伸一助手、宮原研究室の牧一之進助手に心から感謝致します。また、ネットワークプロセッサ IXP1200 に関して助言を頂いた、インテル株式会社の友部昭夫様に心から感謝致します。最後に、ご協力を頂いた村田研究室および宮原研究室の皆様、特に適切な助言をして頂いた山田達也氏に心からお礼を申し上げます。

参考文献

[1] J. Postel, "Transmission Control Protocol," *RFC 793*, Sept. 1981.

[2] J. Padhye, J. Kurose, D. Towsley, and R. Koodli, "A model based TCP-friendly rate control protocol," *UMass-CMPSCI Technical Report TR 98-04*, Oct. 1999.

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

[4] B. Branden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP) – version 1 functional specification," *RFC 2205*, Sept. 1997.

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

[6] Akamai Technologies Inc. available at `http://www.akamai.com/`.

[7] I. Minei and R. Cohen, "High-speed Internet access throgh unidirectional geostationary satellite channels," *IEEE Journal on Selected Areas in Communications vol. 17 no. 2*, 1999.

[8] T. Henderson, "TCP performance over satellite channels," *Technical Report, University of California, Berkeley*, 1999.

[9] A. V. Barkre and B. R. Badrinath, "I-TCP: Indicate TCP for mobile hosts," in *Proceedings of International Conference on Distributed Computing Systems*, pp. 136–143, May 1995.

[10] V. Tsaoussidis and I. Matta, *Open issues on TCP for mobile computing*, vol. 2. Wiley Academic Publishers, Issue 2 ed., Feb. 2002.

[11] C. Barakat, N. Chaher, W. Dabbous, and E. Altman, "Improving TCP/IP over geostationary satellite links," in *Proceedings of IEEE GLOBECOM '99*, Nov. 1999.

[12] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," in *Proceedings of ACM SIGCOMM '96*, pp. 256–269, Aug. 1996.

[13] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*. Reading, Massachusetts: Addison-Wesley, 1994.

[14] H. Balakrishnan and R. H. Katz, "Explicit loss notification and wireless Web performance," in *Proceedings of IEEE GLOBECOM '98 Internet Mini-Conference*, Nov. 1998.

[15] E. Ayanoglu, S. Paul, T. F. LaPorta, K. K. Sabnani, and R. D. Gitlin., "AIRMAIL: A link-layer protocol for wireless networks," *Wireless Networks, vol.1*, pp. 47–60, Feb. 1995.

[16] S. Flolyd, "HighSpeed TCP for large congestion windows," *Internet Draft draft-floyd-tcp-highspeed-01.txt*, Aug. 2002.

[17] V. S. Pai, P. Druschel, and W. Zwaenepoel, "IO-Lite: a unified I/O buffering and caching system," in *Proceedings of the 3rd Symposium on Operating Systems Design and Implementation*, Feb. 1999.

[18] H. Chu, "Zero-copy TCP in Solaris," in *Proceeding of the USENIX 1996 Annual Technical Conference*, Jan. 1996.

[19] J. R. Walsh, "DART: Fast application-level networking via data-copy avoidance," *IEEE Network*, pp. 28–38, July/August 1997.

[20] A. Romanow and S. Bailey, "An overview of RDMA over IP," in *Proceedings of CERN PFLDnet2003*, Feb. 2003.

[21] Intel IXP1200 Network Processor Family. available at `http://www.intel.com/design/network/products/npfamily/ixp1200.htm`.

[22] ENP-2505/2506 Data Sheet. available at http://www.radisys.com/oem_products/ds-page.cfm?productdatasheetsid=105%5.

[23] IXP1200EB Simplified Reference Design. Jan. 2001.

[24] T. Yamada, "Design, implementation, and evaluation of active video-quality adjustment method for hetrogeneuos video multicast," Master's thesis, Osaka University, Feb. 2003.

[25] BlueCat Linux. available at http://www.lynuxworks.com/products/bluecat/bluecat.php3.

[26] BogoMIPS mini-Howto. available at http://www.clifton.nl/gogomips-2.html.

[27] LMbench. available at http://www.bitmover.com/lmbench.

[28] D. Clarke, V. Jacobson, J. Romkey, and H. Salwen, "An analysis of TCP processing overhead," *IEEE Communications Magazine, 27(6):23–29*, June 1989.

[29] G. Hasegawa, T. Terai, T. Okamoto, and M. Murata, "Scalable socket buffer tuning for high performance Web servers," in *Proceedings of IEEE ICNP 2001*, Nov. 2001.

[30] D. Patterson, T. Anderson, N. Cardwell, R. Fromm, K. Keeton, C. Kozyrakis, R. Thomas, and K. Yelick, "A case for intelligient RAM:IRAM," *IEEE Micro*, vol. 17, no .2, pp. 34–44, March/April 1997.

[31] RedHat Linux. available at http://www.redhat.com/.

[32] Intel IXP2400 Network Processor Family. available at http://www.intel.com/design/network/products/npfamily/ixp2400.htm.

[33] IXP1200 Microcode Solutions Library. available at http://www.radisys.com/oem_products/ds-page.cfm?productdatasheetsid=112%3.

[34] E. Yeh, H. Chao, V. Mannern, J. Gervals, and B. Booth, "Introduction to TCP/IP offload engine (TOE)." available at http://www.10gea.org/SP0502IntroToTOE_F.pdf.