リコンフィギャラブルハードウェアによる高速トラフィックモニタの開発とその応用

安田 豊
京都産業大学 理学部
yasuda@cc.kyoto-su.ac.jp

高島 研也
ユーテン・ネットワークス株式会社
kenya@u10networks.com

ストリーミングやIP電話などサービス品質に敏感なサービスが増えつつあり、ユーザ一人一人に提供されるネットワーク・サービスの品質評価の重要性が高まっている。トラフィックモニタリングはそうした分析に適用できる従来からある手法であるが、近年のネットワークの高速化に追随することは容易ではない。本研究ではリコンフィギャラブルハードウェアを用いてGigabitネットワークでワイヤースピードのモニタリングを可能にするシステムを開発した。またこれを実際のネットワークサービスに適用してユーザレベルのサービス品質に注目した分析を行い、その有効性を検証した。

Development and application of the reconfigurable hardware based high speed traffic monitor

Yutaka Yasuda
Kyoto Sangyo University, Faculty of Science
yasuda@cc.kyoto-su.ac.jp

Kenya Takashima
u10 Networks, Inc.
kenya@u10networks.com

The valuation of the quality of service of network application in user basis increases in importance because there are various quality sensitve services are comming up such as video streaming and IP telephony. Traffic monitoring is a method which can analyze them flexibly but it is difficult to apply it to the high speed network. We have developed a monitoring system which is able to monitor at 1 Gigabit wire-speed using reconfigurable hardware technology. And we also analyzed the user level service quality using this on the real network service to validate the effectiveness of this system.

1 はじめに

ユーザに対するサービス品質を維持し、それを正確に把握することはネットワーク・サービスを運用するうえで重要な要件の一つである。特にストリーミングやIP電話などの品質に敏感なサービスが普及しつつあり、その重要度はますます高まっている。

こうした用途ではユーザ一人一人に提供されるサービスの安定性が重要であり、WWWサーバの応答性低下や、ストリーミングサーバからのデータ遅滞といったサービス状況の変動を、ユーザ単位でとらえ得る性能評価法が求められている。

本研究では、リコンフィギャラブルプロセッサを用いたトラフィックモニタを開発し、ネットワーク・サービスの品質評価を行ってその機能を検証した。本稿ではまず開発したモニタの設計と単体テストの結果を示し、これを用いて行う分析手法について述べる。更にこれを実際のネットワーク・サービスに適用して、その有効性を示す。最後にこうした用途にリコンフィギャラブルプロセッサを用いる優位性について議論し、論文をまとめる。

2 背景

トラフィックモニタリングはネットワーク・サービスの評価に従来から用いられている手段の一つである。典型的な例として、NetFlow[1] などに対応したアプライアンス製品や、PC と libpcap[2] 等のソフ

トウェアによるシステムなどがあげられる。

前者はそれが利用可能な機器が一般に高価であり、トラフィックから抽出可能な情報も限定的である。対して PC とソフトウェアによるシステムは安価に実現でき、キャプチャー処理を直接プログラミングできるため多様な分析処理が可能であるが、その処理能力が低く Gigabit クラスのトラフィックに適用するのは難しいという問題がある。

サービス品質評価の需要が今後増し、アプリケーションの多様化にともなって分析手法も多様化すると考えると、比較的安価で、そうした分析処理を容易に実装できるシステムが必要になる。

一方、リコンフィギャラブルハードウェアの能力が高まっており、これによって Gigabit クラスの処理が可能になりつつある。

そこで本研究ではリコンフィギャラブルプロセッサを用いたインテリジェントなネットワークインタフェイス・ボードを開発し、モニタリングや分析処理の一部をハードウェアに移して、残った分析や可視化の作業をホストとなる PC で行うシステムを構築した。

リコンフィギャラブルハードウェア技術をベースにモニタリング処理能力の向上とホスト CPU のオフロードを実現する例には他に、DAG カード [4] を用いた研究 [5], [6] などがあるが、FPGA を利用している点で本研究とは異なる。

本研究でリコンフィギャラブルプロセッサを用いた理由は、ハードウェア上に実装した機能の修正や追加の容易さを重要視した結果である。この点については、まず開発したシステムについて述べた後に改めて 5 章で論じる。

3 モニタリング装置

3.1 設計

目標は Gigabit ネットワークでロストのないキャプチャ能力を安価に実現することである。そのために装置は標準的な PC と、新規に開発したインテリジェントにパケット処理を行うネットワークインタフェイス・ボード（図 1、以後 GigaPcap と呼ぶ）によって構成する。GigaPcap は 1000BaseT インタフェイスを 2 ポート搭載し、PCI-X バスでホスト PC に接続される。

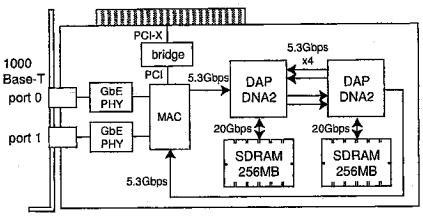

図 1: GigaPcap block diagram

GigaPcap はリコンフィギャラブルプロセッサ (IPFlex 社製 DAPDNA-2[3]) を 2 チップ搭載する。このプロセッサは 32bit アーキテクチャの要素プロセッサ (PE) を 376 個 (うち演算器が 168 個) 内蔵し、外部から PE 間の配線とソフトウェアを与えることができる。動作周波数は 166MHz である。

このプロセッサで前処理を行うことでホストシステムの処理負荷を軽減する。具体的には libpcap の機能を部分的に実装した (図 2)。

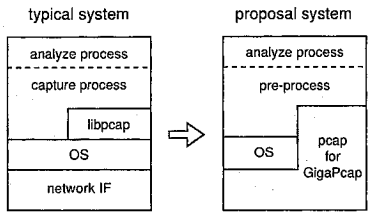

図 2: libpcap の実装

GigaPcap 上に実装した機能には snapshot 処理、タイムスタンプ付加などがある。フィルタ機能は未実装[1]である。またホストへのデータ転送はパケットごとではなく一定量ごとにまとめて行う。

これらによって libpcap を利用した簡易なプログラミングと、高性能なモニタリングの両立が可能となり、ホストシステムがそれ以降の分析処理等に必要とする処理能力を残すことができた。

3.2 タイムスタンプ精度

モニタリングによる性能測定には精度の高いタイムスタンプが必要である。しかし高速ネットワークにおけるモニタリングではシステム処理能力の限界によってその精度が低下することが Jin[7] らによっ

[1]libpcap とは非互換の比較的単純なフィルタを実装した。

て指摘されている。GigaPcap ではボード側でタイムスタンプ処理を行うことでこの問題を解決する。

GigaPcap ではプロセッサの前段に FPGA で実装された MAC 処理部があり、これがパケットの末尾を受け取るとパケットデータをバッファからプロセッサに転送しはじめる。タイムスタンプはその転送完了時に付加されるため、この転送遅延が主たる誤差となる。プロセッサへの転送は 6ns ごとに 4 バイトずつ行われる (約 5.3Gbps) ため、遅延は 1.5KB パケットでの約 2.3 μ秒が最大である。

なお特別なネットワークインタフェイスを利用せずに 1 μ秒またはそれ以下の精度でのタイムスタンプを実現する研究もあるが [8], [9]、その精度はホスト OS 等の負荷の影響を受けてしまう。前者はリアルタイム OS を用いて影響を減じ、後者は逆に影響を受けた回のサンプルを破棄して精度を維持している。

GigaPcap のタイムスタンプ精度はそれ自身およびホスト側の負荷状況に左右されず、ホスト OS の選択や分析アプリケーションの実行負荷などに対する制約が低い点で優れている。

3.3 性能評価

まず GigaPcap の性能評価を行った。評価は二台の PC にそれぞれ GigaPcap を搭載し、一方でパケットを生成して他方に送り、受信結果を見ることで行う。

まずパケット生成側の安定度を確認するために、最も生成時の負荷が高くなる 64 バイト長のパケットについて一定のレートで 1000 パケット送り、受信側では専用プログラムを用いてその到着時刻を GigaPcap のクロック (6ns) で記録する。

最短のパケット送出時間はプリアンブル+最小パケットの送出時間に必須ギャップを加えた 112 クロックであるため、そうなるまで送出間隔を 1 クロックずつ短くして測定する。ホスト OS の影響を遮断するため、全ての処理は GigaPcap 上で行った。

結果は以下の通りである。

- 全ての送出間隔において 99.9% 以上のパケットが送出時と同一のクロック間隔で受信された。
- それ以外も全て±1 クロックにおさまっている。

つまりほぼ安定して一定速度の連続パケットを生成でき、かつこれを安定して受信できていると言える。

次にこの送出機構を用いて GigaPcap 用 pcap ライブラリを用いたキャプチャー性能を調べる。試験は、以下のパケット長と送出レートの組み合わせについて 1000 万パケットを連続して送出し、pcap ライブラリを通してホスト側のアプリケーションプログラムがロストなく受信できたかどうかで調べた。

- パケット長 : 64, 200, 500, 1000, 1518 バイト
- 送出レート : 50, 200, 500, 700, 1000Mbps

libpcap の snaplen、つまりパケット先頭からの切り落としは設定せず、全ての試行で全パケットデータをホスト側に渡している。snaplen を短く設定するとホスト側の負荷は相対的に下がるため、これが最も厳しい条件となる。なお、送出レートはプリアンブル、パケットデータ、最小フレーム間ギャップを合わせた値を用いて算出している。

実験の結果、全てのパケット長と送出レートについてロストなしに受信できることが確認できた。試験機は DELL PowerEdge 2850 (2.8GHz Xeon x1, 1GB Mem., E7520 chipset) に RedHat Enterprise Linux 4 (kernel 2.6.9) を導入したものである。

4 実験

モニタリング装置に分析処理を加えたシステムを構築し、実際に運用されている Windows Media Services 9 (WMS9) による野球中継のライブ・ストリーミング・サービスを対象に実験を行った。対象システムは負荷分散装置を用いて複数サーバで構成されているが、うち一台について計測する機会を得た。

まず全体トラフィックを図 3 に示すが、20 時 15 分頃に合計転送速度の落ち込みが確認できる。

以下に、個々のユーザ単位のトラフィック情報を分析することで、この時間帯に発生したサービス状況の変化をより具体的に把握することを目的に分析を試みる。

4.1 情報の集約

高速ネットワークにおけるトラフィック分析では、処理データ量削減のための前処理が重要である。本研究ではサンプリング等はせず、全てのパケットを対象に表 1 の項目を対象にした集約処理を行う。

すなわちトラフィックは、L3 プロトコルタイプ、IP アドレス組、TCP/UDP についてはそのポート組ごとについて、一定単位時間ごとに通過したパケット数と転送データ量のカウントに集約される。

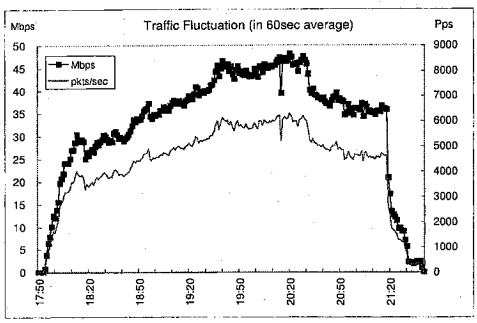

図 3: 全体トラフィック

	Bytes
protocol type (Layer 3)	1
IP address pair	4 * 2
port number pair	2 * 2
packet count	4
data length	8
lifetime (microsec)	4

表 1: 集約フォーマット

本稿ではこの集約単位をフローと呼ぶ。転送データ量は TCP/UDP についてはそのヘッダを除いたデータ部、それ以外では IP ヘッダ以降の長さで数える。パケット長を取らずこのようにしたのは今回の分析ではストリーミングサービスの安定性に注目したためである。

また、単位時間ごとの最初と最後のパケット通過時刻から、各フローの存在時間をマイクロ秒単位で算出する。図 4 は時刻 t から u 時間ごとに集約単位をとった例である。この場合 Flow A, B は表 1 のフォーマットで 4 組のデータに集約され、Flow A の存在期間は $a1 + u + a2$、Flow B は b となる。

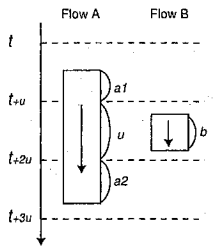

図 4: フロー存在時間の算出

フローの開始、終了は単位時間内に登場した最初のパケットと最終パケットの通過時刻差を用いるが、トラフィックがその前後の時間帯にまたがっている場合は補正を行う。例えば Flow A の最初のブロックでは終了時刻を $t + u$ に、中間のブロックでは開始、終了時刻を $t + u$ と $t + 2u$ にする。

誤差の少ない転送速度の算出にはこうした生存期間情報が重要である。例えばフローの開始、終了が単位時間中のどのあたりになるかは全くランダムであるから、期間情報なしで得た転送速度はフローの先頭・末尾時間帯においては実質的に意味がなくなる。その精度を高めるために集約期間を短時間にしたのでは本来の目的であったデータ量削減ができない。

今回の実験では 10 秒単位、かつ往復のフローを同一視して集約した結果、パケット先頭から 100 バイトで snap したキャプチャデータ 4.0G バイトを 3.1M バイトまで、1/1000 以下に集約することができた。

4.2 可視化

集約されたトラフィックデータを元に、各フローごとの転送速度の変化について可視化を試みた。

図 5 は図 3 で問題となった 20 時 15 分頃に、各フローの転送速度がどのように変化したかを可視化したものである。横軸は時間の経過を示し、左から右に流れる。縦軸は 10 秒ごとの平均転送速度であり、各フローごとに速度に応じて打点し、分布を見る[2]。

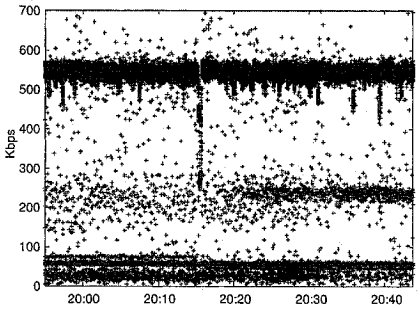

図 5: 転送速度分布の時間変化

なお、WMS9 によるサービスでは数種のプロトコルが混在するが、今回の可視化ではプロトコルによる差異が大きく出なかったため混在させたままで

[2]実験時は図 6 と同様のディスプレイ向けグラフィクスで可視化した。これは印刷により適した形式で再プロットしたもの。

描画した。内訳はフロー単位で数えて RTSP (TCP) 53%, RTSP (UDP) 17%, MMS (TCP) 9%, HTTP (TCP) 8%, その他 (不明含む) であった。

全体を通して多くのフローが 550Kbps 程度にあったが、20 時 15 分頃にほぼ全てが一時的に速度を下げ、20 秒 (2 単位時間) 後に戻っている。ただしそれ以降、一段低い 250 Kbps 付近に集まるフロー群が生じていることがわかる。

次にフローごとの転送速度を色で区別できるように可視化した結果を示す (図 6) 。

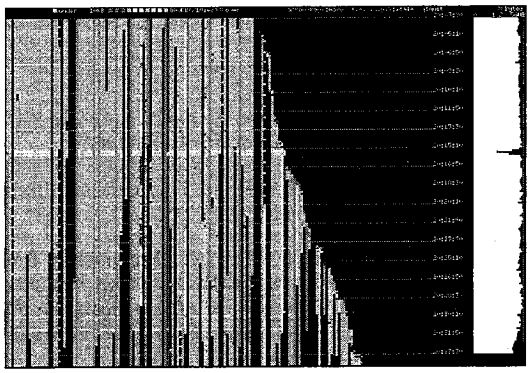

図 6: フローごとの転送速度変化

時間は上から下に流れ、各フローは転送速度に応じた色 (低速が寒色、高速が暖色) で縦に描画される。新しく接続されたフローほど右に並んでいる。右端のグラフは全体トラフィックを示し、20 時 15 分頃の落ち込みと同期して全てのフローの転送速度が落ち、幾つかのフローは停止し、それ以降の接続には低速なものが比較的多く含まれるようになった状況がわかる。

WMS9 には、利用可能な帯域に応じて最適な転送速度を選択する機能がある。一時的にサービス能力が低下した後に接続されたクライアントには、充分な転送速度でサービスがされなかったため、一段低い速度が多く選択されたと推測される。

もちろん以上の結果だけでこの時間帯に発生した問題の原因を突き止めるのは困難であるが、本研究における計測・可視化の目的である各ユーザに対するサービス状況の変動をより具体的に把握することはできた。

5 考察

5.1 実装すべき前処理の性質

本研究はトラフィック分析処理の実行負担を、部分的にホスト側のソフトウェアからインテリジェントなハードウェアに移すことによって Gigabit クラスの高速化を実現するものである。

しかしさらなる高速化、ホスト側の負担減のためには、より積極的にアプリケーションに近い部分を移す必要がある。本研究の例では 4.1 で示した集約処理までを GigaPcap に移すことが考えられる (図 7)。

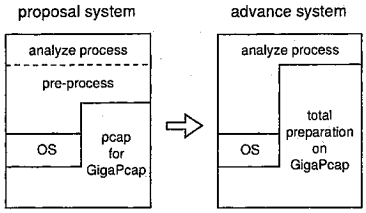

図 7: より積極的な前処理の実装

ところが前処理の内容は後段で行う分析方針に依存性がある。例えばフローの存在期間は 4.1 に示した方法ではなく、TCP の SYN, FIN 等のフラグで測る方法が有り得るが、それが適切な場合とそうでない場合がある。

HTTP のように、サーバ・クライアント双方の Keep Alive 対応の有無でフラグに基づいたトラフィック存在期間の意味が大きく異なるサービスも存在する。

今回対象としたストリーミングサービスでは、ユーザが再生を一時停止したために、少量のパケットが間欠的に送られる状態になったセッションが複数存在した。これを断続的なトラフィックとして検出する方が良いか、単位時間あたりの転送速度に平均化すべきかは分析方針の問題であり一概には決められない。

すなわち分析対象となったサービスおよび分析の目的によって前処理の内容は異なり、何らか共通の手法ですべてのサービスに充分に対応できることは期待できない。むしろより的確なサービス品質評価のために、分析対象に合わせた前処理を柔軟に実装・導入できることが望ましい。

本研究でリコンフィギャラブルハードウェア技術

を利用したのは、こうした修正や機能追加を目的としている。

5.2 DAPDNA-2 での開発

リコンフィギャラブルハードウェアによる開発としては FPGA による例が多いが、本研究ではリコンフィギャラブルプロセッサである DAPDNA-2 を用いた。

その理由は、DAPDNA-2 が 5.1 に示した、より高度な前処理に必要な高い演算能力をもつことと、FPGA に較べて後段に位置する分析アプリケーションが要求する多様な機能追加や、頻繁な修正を行うのにより適した性質をもっているためである。

DAPDNA-2 の開発環境には PE 間の接続とインストラクションを GUI によって直接指示する DNA Designer [10] を利用している。DAPDNA-2 は内部が完全同期式回路で構成されているため、この開発環境において処理のスループット及び遅延を厳密に保証することができる。

これはあらかじめワイヤースピードといった達成すべき処理能力が明らかな状況でフィルタや集約処理などのアルゴリズムを検討する際には極めて有効であり、性能保証を求められる本研究には重要である。

また、機能ロジックを追加する際に元のロジックに影響が出ないため機能の追加が容易である。これも個々のアプリケーションの要求に応じてハードウェアに機能を載せていこうとする本研究に適している。

もし実装すべきロジックが 1 チップに収まらない場合、DAPDNA-2 では複数チップによる並列パイプライン処理が可能である。そのために GigaPcap には二つの DAPDNA-2 が搭載されており、これを 5.3Gbps の高速インタフェイス 4 チャネルで接続している。

本研究で実装した処理はほぼ 1 チップに収まっており、より高度な前処理を追加導入するのに充分な余裕を残している。4.1 に示したような集約処理に必要となるメモリもプロセッサごとに 256M バイト搭載しており、今後の機能追加にそなえている。

6 おわりに

本研究では、リコンフィギャラブルプロセッサによって Gigabit ネットワークのワイヤースピードを処理可能なモニタリング・システムを開発し、これに libpcap を対応させた。またこれを実際のネットワークサービスに適用して分析を行い、各ユーザに生じたサービス品質の変化を明らかにすることで、その有効性を検証した。

そしてさらなる高速化のためにより積極的にハードウェアを利用する手法について検討し、そこでは従来ソフトウェアで実現されていたような柔軟性がハードウェアにも要求されることを示し、リコンフィギャラブルプロセッサの優位性について確認した。

今後の研究として、libpcap 互換フィルタの実装、集約処理のボード上への移行など、より積極的なハードウェア上への機能実装を進めていく予定である。

参考文献

[1] B. Claise, Ed. : Cisco Systems NetFlow Services Export Version 9, RFC3954, 2004

[2] S. McCanne, C. Leres and V. Jacobson : libpcap, http://ee.lbl.gov/

[3] T. SATO : DAPDNA-2 : A Dynamically Reconfigurable Procesor with 376 32-bit Processing Elements, *Proceedings of A Symposium on High Performance Chips 17 (HOTCHIPS17)*, 2005

[4] ENDACE, http://www.endace.com/

[5] J. Coppens, et al. : SCAMPI - A Scaleable Monitoring Platform for the Internet, *Proceedings of the 2nd International Workshop on Inter-Domain Performance and Simulation (IPS 2004)*, 2004

[6] A. Paszor and D. Veitch : High Precision Active Probing for Internet Measurement, *Proceedings of INET 2001*, 2001

[7] G. Jin and B.L. Tierney : System Capability Effect on Algorithms for Network Bandwidth Measurement, *Proceedings of Internet Measurement Conference 2003*, 2003

[8] A. Pasztor and D. Veitch : PC Based Precision Timing Without GPS, *Proceedings of ACM SIGMETRICS 2002*, 2002

[9] 町澤 朗彦, 北口 善明 : 割り込みハンドラと高精度 PC によるソフトウェアタイムスタンプの精度改善, 電子情報通信学会論文誌 B Vol.J87-B No.10, 2004

[10] 末吉 敏則, 天野 英晴 編著 : リコンフィギャラブルシステム, オーム社, pp.187-, 2005