社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2003－ＱＡＩ－8　(7)
2003／7／7

分散型リアルタイムフロー計測ツールの設計、実装と性能評価について

北辻　佳憲　　山崎　克之

株式会社 KDDI研究所　〒356–8502 埼玉県上福岡市大原 2–1–15
E-mail: {kitaji,yamazaki}@kddilabs.jp

あらまし　最近のインターネットの多様なサービス提供に対する性能監視や、サービス不能化攻撃の監視・同定・追跡は、MRTGなどのIP層のトラヒック量の監視だけでは実現することが難しく、トランスポート層およびその上位のアプリケーションのフロー計測が有効である。しかし、フローの同定処理の複雑さや識別されるフローが多量に検出されることから、高速な回線を対象とするハードウェア計測装置の開発は高価になる。本稿では、トラヒックを分散してリアルタイムにフロー計測を行うための要求仕様を明らかにし、仕様を満たすソフトウェアツールの方式を提案する。更に、同ツールをPC-UNIX上に実装し、性能評価を行ったのでその結果を報告する。
キーワード　リアルタイムフロー計測，計測ツール

Design, Implementation and performance evaluation of a distributed real-time flow measurement tool

Yoshinori KITATSUJI and Katsuyuki YAMAZAKI

KDDI R&D Laboratories, Inc.　Ohara 2–1–15, Kamifukuoka-shi, Saitama, 356–8502 Japan
E-mail: {kitaji,yamazaki}@kddilabs.jp

Abstract　It is getting more difficult to monitor the multiple service classes provided to various customers or to detect and/or to trace the Daniel of Service attacks with using only tools showing graphs of whole IP layer traffic of links like MRTG or checking counters of router interfaces. It is much useful to use a flow visualization tool based on any portion of headers between the network layer and applications transported by the transport layer at this kind problem. However, the development of hardware of such flow measurement tools enabling to measure high speed links costs much because of the its complexity of flow detection process and a number of flows detected in a short time passing through such high speed links. We discuss of specification of the real-time flow measurement tool which consist of multiple capture devices, a manager device and a user interface device, and propose its architecture in this paper. And then we report the evaluation of the performance of a prototype of proposed real-time flow measurement tool developed on PC-UNIX.
Key words　Realtime flow analysis, Measurement tool

1. はじめに

最近のインターネットの多様なサービス提供のための性能監視や、サービス不能化攻撃 (DoS アタック) の監視・同定・追跡は、MRTG [1] などのトラヒック量の監視だけでは実現することが難しく、トランスポート層およびその上位のアプリケーションのフロー計測が有効である。しかし、フローの同定処理の複雑さや識別されるフローが多量に検出されることから、高速な回線を対象とする計測装置の開発は高価になる。そこで、本稿ではトラヒックを分散してリアルタイムにフロー計測を行なうソフトウェアツールの要求仕様を明らかにし、その方式を提案する。特に、柔軟なフロー定義と重複するパターンマッチングの処理を省略するフロー定義のデータ構造や、複数のキャプチャ装置と管理装置の連携について提案する。更に、同ツールをPC-UNIX上に実装し、性能評価と実ネットワークを対象とした実証を行ったので、その結果を報告する。

2. フロー計測の必要性と課題

一般に、ISP等のバックボーンで多量のトラヒックを交換する環境では、サービス不能化攻撃 (DoS アタック) に対して、下記の (1) から (3) の手順を繰り返して攻撃の被害を食い止める方法がとられる。

(1) 特定のパターンを持ったトラヒックの検出
(2) トラヒックのフィルタや BGP 経路情報の交換の制限
(3) トラヒック変化の確認

一連の作業で、(1) および (3) では、MRTG やルータのインタフェースのカウンタなどが用いられるが、トラヒックの総量の変化という特徴だけから DoS アタックの判定を行うには、運用の経験的な感に頼る部分も少くない。このような状況では、IP アドレスあるいはトランスポート層のポートに限らず、トランスポート層を含む上位のアプリケーションデータを対象に、特定のパターンを持ったフローをリアルタイムに抽出し、その変化を確認できることが望ましい。

フローは、送信元および宛先の IP アドレスおよびポート番号が一致するパケット列のうち、前後のパケットが特定の時間以内に継続するパケットの集合と定義されるが [3]、本稿では、特定の位置のバイト列が同一なパケットの集合と定義する。そうすることで、上述のトラヒックの抽出をフロー計測の一つとして捉えることができる。フロー計測は、個々のフローの長さ (パケット数または全パケットのバイト数) および継続時間 (開始と終了パケットのタイムスタンプの時刻差) を測定値としてフローの統計値を算出し、測定点を流れるトラヒックの特性を明らかにする。フロー計測の処理は、トラヒックのキャプチャ、フローの同定、統計値の算出、データ保存の処理から構成される。高速な回線におけるフロー計測では、ツールが十分高速でなければならず、高価なハードウェアが必要となる。

3. 設　　計

前章の課題をもとに、安価な PC-UNIX を用いて、複数の装置から構成されるフロー計測装置の設計について議論する。

3.1 要求仕様

フロー計測の基本処理の中で、処理負荷が大きいフローのキャプチャや同定処理を複数の装置で分散して行うフロー計測装置の要求仕様を検討する。

[装置の構成] フローの同定処理を分散して行うために、図 1 のように、システムの構成を分散装置 (Distribute device)、キャプチャ装置 (Capture device)、管理装置 (Manager device)、ユーザインタフェース装置 (User Interface) に分離し、パケットのキャプチャからフローの同定までの処理を複数の装置で動作させることにする。キャプチャ装置は、計測するトラヒック速度に合わせて柔軟に装置数を変更できることが望ましい。さらに、ツールの運用の負担を小さく抑えるためには、フローの定義は管理装置が管理し、キャプチャ装置に通知できる機構を備えるべきである。装置構成では、キャプチャ装置の追加／削減や、フロー定義の整合性を維持する装置間の連携が重要となる。

[分散方法] 高速な回線のトラヒックを分散する場合、パケットの特定のバターに依存した分散は、片寄りを起す可能性があり、フローの同定処理を分散化する効果が発揮されない恐れがある。できるだけ均等にトラヒックを分散することで、より高速なトラヒックの計測が可能であると仮定すれば、処理負荷が小さい round robin 方式が分散に適してる。キャプチャ装置は、全てのフロー定義を登録して、分散装置の分散方法に依存せずに動作可能なことが望ましい。

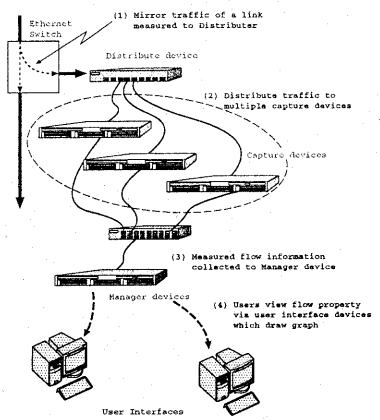

図 1　リアルタイムフロー計測ツールの基本構成
Fig. 1　Basic architecture of realtime flow measurement tool

[フローの区別および継続時間]: 複数のキャプチャ装置の一つでフローの終了が判定されたとしても、全キャプチャ装置の観点からは、同じフローが継続中と判定される場合がある。そのため、フローの開始／終了時刻およびフロー継続の判定を管理装置で行えるように、管理装置への計測値の通知の際に、前回の通知後からの最初と最後のパケットのタイムスタンプを通知する必要がある。

[時刻同期]: フローの開始/終了時刻および継続時間はキャプチャ装置での受信パケットのタイムスタンプが基準となるため、複数のキャプチャ装置での時刻の同期は必須となる。NTP による時刻の同期では stratum2 における時刻誤差が数ミリ秒前後であるのに対して、GPS による時刻制度は数マイクロ秒となることが報告されている [2]。計測点によっては GPS アンテナの設置が難しくなる場合も考えられるが、高速な回線のトラヒックを計測するためには、GPS または同等の精度を持つシステムとの時刻同期が望ましい。

[フローの定義方法]: リアルタイムにフロー計測を行なう場合、運用状況によって計測するフローを自由に選択できることが望ましい。フローのパターンは利用者が自由に定義でき、定義の更新の際、装置を停止することなく更新可能なことが望ましい。また、高速なパターンマッチングを行うためには重複するマッチングを省略できる定義方法が有効である。

[計測落ちの検出]: フローの同定に多数の複雑なパターンマッチングを行えば、計測性能が劣化することは否めない。しかし、性能以上のトラヒックをキャプチャしたために計測落ちが発生したことは検出できなければならない [3]。さらに、計測落ちが検出された場合の動作の継続または停止は、利用者の判断に委ねられるべきであり、どちらの動作も選択できることが望ましい。

3.2 システム構成

前節の要求仕様をもとに、筆者等が提案する、汎用 PC-UNIX による分散型リアルタイムフロー計測ツールの動作の概要を図 2 に示す。

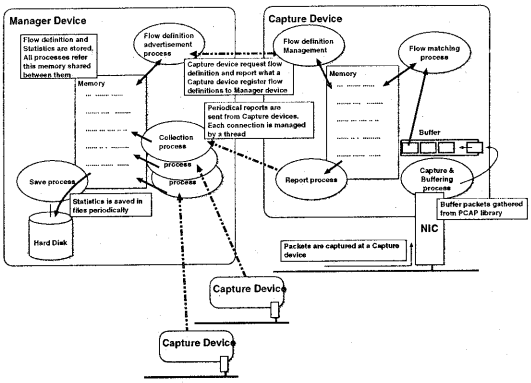

図 2　フロー計測ツールの動作と連係の概要
Fig. 2　Relation of devices and function

イーサネットスイッチや光カプラ装置でコピーされたトラヒックは分散装置に送られ、複数のキャプチャ装置へ分散される。本稿では分散装置は汎用の製品を用いるものとする。

キャプチャ装置では、以下の 4 つの処理がスレッドで並列に動作する。

- pcap ライブラリを核とするパケットバッファ処理
- フローの同定および統計値を更新するフロー同定処理
- 管理装置とフロー定義のやり取りを行う定義更新処理
- 定期的な計測値の通知を行う通知処理

パケットバッファ処理の導入は、瞬間的なバーストトラヒックを吸収して、キャプチャ落ちを抑制することを目的としている。pcap ライブラリの利用は、多くの OS での実装・移植を容易にするためであるが、その他に、pcap ライブラリがカーネルの持つ受信パケットカウンタの参照方法を提供する。これを用いれば、アプリケーションレベルのパケットロスの検知がし易くなる。

キャプチャ装置では、すべてのフローを判定できるように、同一のフロー定義が通知される。定義更新処理はフロー定義の通知を受けると、パケットバッファ処理、フロー同定処理、通知処理を、スレッドによって派生させ、3 つの処理の間でフロー定義および計測値に差異が起こらないよう、開始および終了を管理する。

管理装置では、以下の 3 つの処理がスレッドで並列に動作する。

- フロー定義をキャプチャ装置へ通知する定義通知処理
- 計測値を収集する収集処理
- 統計値をファイルへ保存する保存処理

管理装置の動作は、定義通知処理が収集処理および保存処理を、スレッドによって派生させることとし、3つの処理の間でフローの定義および計測値に矛盾が起こらないよう、開始および終了を管理する。定義通知処理は、フローの定義要求および確認応答に従ってフロー定義をキャプチャ処理に通知し、新しいキャプチャ装置からの定義要求を受けるたびに、各キャプチャ処理専用の収集処理をスレッドによって派生させる。

表 1　ビットパターンを特定する要素
Table 1　Elements specifing bit pattern

識別子	複数のビットパターンを用いるフロー定義でパターン間の参照に用いる
位置	パケットの先頭からのビットパターンの開始位置
大きさ	連続するビットパターンのサイズ
マスク	バイト単位のパターンから判定に有効なビットを特定する
パターン最小値	範囲を指定する場合の最小値
パターン最大値	範囲を指定する場合の最大値
子パターン識別子	不連続なビットパターンの関係を示すリスト

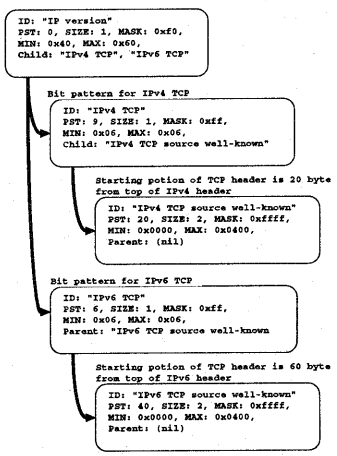

図 3　TCP の well-known ポート別フローの定義の例
Fig. 3　Example of flow definition for TCP well-known port flows

3.3　フローの定義手法とデータ構造

本ツールでは、ユーザが定義するバイト単位のビットパターンに一致するパケット列を、フローと定義する。ビットパターンは任意の位置の任意のサイズで不連続であっても良い。本ツールではフローの定義を、表 1 の要素で構成されるビットパターンの集合で与えるものとする。ビットパターンの定義にはマスクを用いて、バイト列の中の有効なビットを定義する。

例えば、TCP の well-known ポートの分布を計測するような場合、図 3 のようにビットパターンの連鎖でフロー定義を構成することができる。

以上のように、フロー定義を複数のビットパターンで構成することで、下記の利点があり、フロー同定処理の負荷低減が期待できる。

- 複数のフロー定義の重複するパターンのマッチングを 1 度のマッチングに省略できる。
- 不連続なビットパターンを持つフロー定義のマッチングにおいて、途中までしか一致しない場合には、一致しなくなったところからそれ以上のマッチングを行わずに済む。

3.4　フローの検索

前節で提案したフロー定義にパターンの範囲（パターンの最

小値と最大値）を定義したことで、1つのフロー定義で複数のフローを検出することが可能である。1つのフロー定義に対して、検出される複数のフローの1つ1つは、最後に用いたパターンマッチングのフィールドの値によって区別できる。このフィールドの値をフロー識別子と定義する。

パケットがフロー定義にマッチすると、それまでに登録されている複数のフロー識別子の中から、同一の識別子を持つカウンタを検索し、そのカウンタの値を増加させる。フロー識別子が登録されていなければ、新たにフロー識別子を登録する。フロー定義の範囲が大きければ、登録されるフロー識別子の数が多くなり、検索時間の増加を引き起こす恐れがある。固定長の識別子をキーとして検索する場合、2分探索が高速だが、フロー識別子を登録して次第に探索空間が拡大する点について考慮が必要である。そのため、本稿では平衡2分木を構築して2分探索を行う。[4]

4. 実装と性能評価

前章で設計した分散型リアルタイムフロー計測ツールについて、PC-UNIX に実装し、下記性能の計測を行った。

- フロー定義数に対する性能
- フローパターンの長さに対する性能
- フロー識別子の数に対する性能
- キャプチャ装置における基本処理の負荷

4.1 実　　装

計測環境を図4に示す。計測は図中のトラヒック生成装置 (Generater) から1台のキャプチャ装置 (Capture Device) にトラヒックを送出して、キャプチャ落ちを起こしたパケット数を数えることとした。キャプチャ装置は、実環境での利用を想定して、前章で示したように、管理装置 (Manager Device) からのフロー定義の取得と、管理装置への定期的な計測値の通知を行わせた。Generater, Capture Device、Manager Device には、全て表2の仕様の PC(DOS/V 機) を用いた。

4.2 フロー定義数と性能の関係

フロー定義の数に対するキャプチャ性能を調べるため、1から64までのフロー定義 (単一パターン) について、500 packet/sec から 33K packet/sec のトラヒックをキャプチャした場合のキャプチャ落ちを計測した。図5にそれぞれのキャプチャ落ちの計測結果を示す。試験は、管理装置を用いてキャプチャ装置にフロー定義を登録し、フロー定義と同じパターンを持ったパケットをトラヒック生成装置からキャプチャ装置へ送信する。フローが複数定義される場合は、生成トラヒックのパターンを round robin 方式で与える。パターンの長さは4バイトとして、完全一致 (単一フローを検知し、その他を不一致とする) のフロー定義を与えた。図5は、横軸が、キャプチャ装置へ転送されたトラヒック速度を、縦軸が1秒当りのキャプチャ落ちを、奥行きの軸がフロー定義の数を示している。図から、フロー定義の数と性能の関係がほぼ反比例することが判る。フロー定義数が1つの場合で最大 27K packet/sec の性能を観測した。

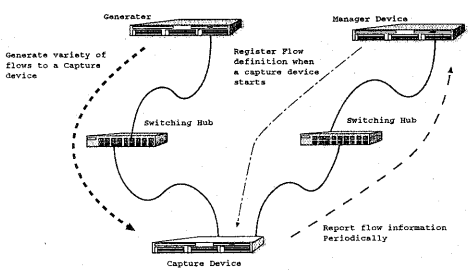

図4　性能計測の構成

Fig. 4　Configuration of performance evaluation

表2　性能計測に用いた PC の仕様

Table 2　Specification of PCs used for Evaluation

CPU	Xeon 2.8GHz
Memory	2GB
HDD	73GB
Bus	PCI-X (64bit, 133MHz)
Network interface	2 10/100Base-TX
Operating system	RedHat 9 linux kernel 2.4.20

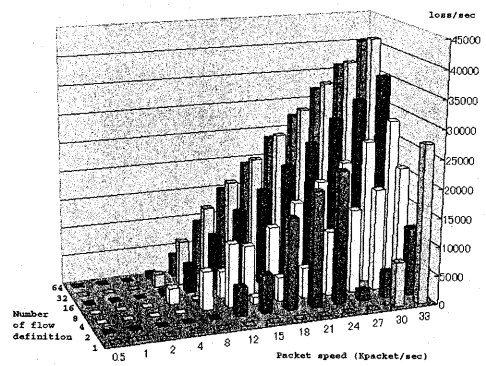

図5　フロー定義数に対するキャプチャ装置の性能

Fig. 5　Capture loss depending on the number of flow definition

4.3 フローパターンの長さと性能の関係

フロー定義のパターンの長さに対するキャプチャ性能を調べるため、1から32バイトの長さのパターンのそれぞれについて、4K packet/sec から 30K packet/sec のトラヒックをキャプチャした場合の、キャプチャ落ちを計測した。図6にそれぞれのキャプチャ落ちの数を示す。試験は、前節と同様の方法でキャプチャ装置にフロー定義を登録し、フロー定義と同じパターンを持ったパケットを定義の種類だけ round robin に送信する。フロー定義の数は1つとした。図6は、横軸がキャプチャ装置へ転送されたトラヒック速度を、縦軸が1秒当りのキャプチャ落ちを、奥行きの軸がパターンの長さを示している。

4.4 フロー定義の範囲の大きさと性能の関係

フロー定義の範囲の大きさに対するキャプチャ性能を調べるため、1つのフロー定義に1から512のフロー識別子を生成するトラヒックのそれぞれについて、1K packet/sec から 30K packet/sec のトラヒックをキャプチャした場合の、キャプチャ

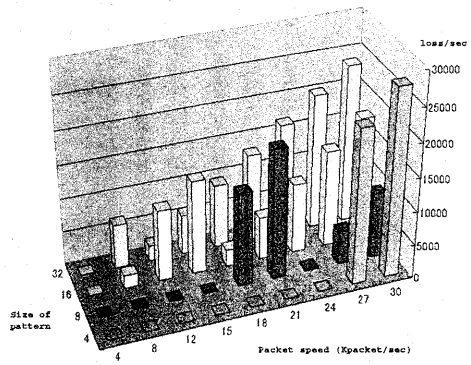

図 6　パターンの長さに対するキャプチャ装置の性能
Fig. 6　Capture loss depending on the length of pattern

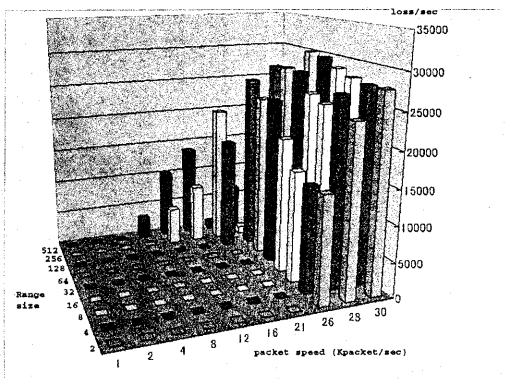

図 7　範囲の大きさに対するキャプチャ装置の性能
Fig. 7　Capture loss depending on the range of flow definition

落ちを計測した。図 7 にそれぞれのキャプチャ落ちの数を示す。フロー定義の与え方は前の 2 つの節と同様で、トラヒック生成については、フロー定義の範囲に収まるパターンを持ったパケットを round robin に送信する。パターンの長さは 4byte とした。図 7 は、横軸がキャプチャ装置へ転送されたトラヒック速度を、縦軸が 1 秒当りのキャプチャ落ちを、奥行きの軸が定義の範囲の大きさ (フロー数) を示している。フロー定義の範囲の大きさが 2 から 64 の範囲では性能の差が現れておらず、最大で 21K packet/s の性能を観測した。

4.5　処理負荷の計測

キャプチャ装置でのフローの検出および登録にかかる処理の負荷を計測するため、

- 受信パケットのバッファ入力処理
- パターンマッチング処理の全パターンの和
- フロー識別子検出後の登録フロー検索処理

に要した CPU クロック数を計測した。1 から 32 のそれぞれのフローパターンに完全一致させる場合の各処理にかかる CPU クロック数を図 8 に、1 パターンで 1 から 512 のフローを検出する範囲を指定した場合の各処理にかかる CPU クロック数を

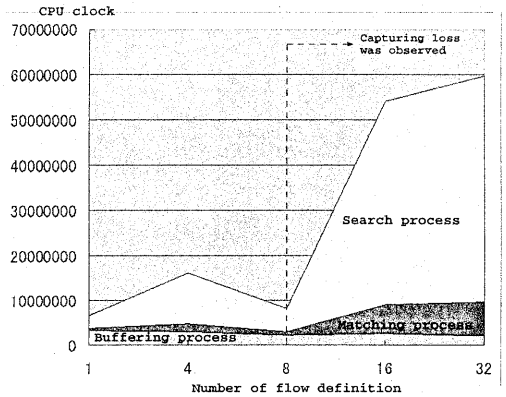

図 8　フロー定義の数に対する処理負荷
Fig. 8　CPU clocks depending on the number of flow definition

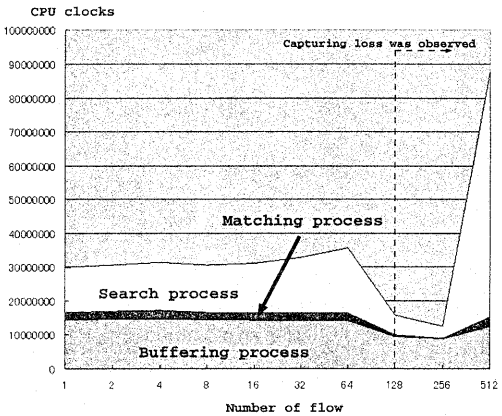

図 9　フロー数に対する処理負荷
Fig. 9　CPU clocks depending on the number of flow

図 9 に示す。図 8 はトラヒック速度を 4K packet/sec とし、横軸をフロー定義の数、縦軸を CPU クロック数とした。図 9 はトラヒック速度を 20K packet/sec とし、横軸を 1 つのフロー定義の範囲、縦軸を CPU クロック数とした。両図は 10 回の計測の平均値を示している。

図に示すように、CPU クロックが減少したパターン数またはフロー数からキャプチャ落ちを観測した。定義数に対する処理負荷では、トラヒック生成速度を 2K packet/sec または 10K packet/se とした場合でも、同様の現象を観測した。

5.　考　　察

5.1　性能評価に対する考察

前章の図 5、6、7 から、フロー定義のパターン数、パターンの大きさ、登録されるフロー識別子の数、の順に性能に影響を与えることが分る。3 つ性能計測から、パターンの長さの減少よりも、フロー定義のパターン数を減らす方が、性能改善により効果があると言える。

本稿で提案するパターンの連結によるフロー定義は、多数のフローを定義する状況で、全体のフロー定義数を削減できる柔

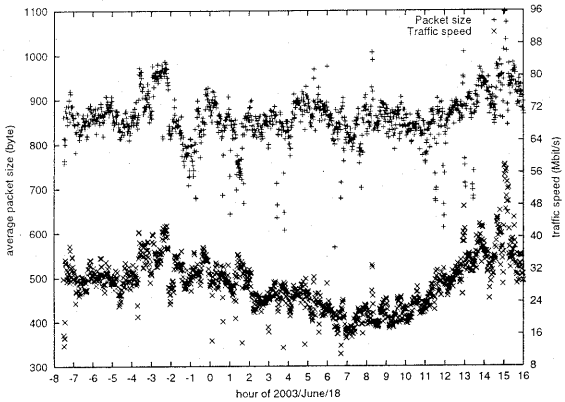

図 10 APAN と WIDE の間を流れるトラヒックのパケットサイズ
Fig. 10 Packet size of traffic passing between APAN and WIDE

軟な方式であるため、多重効果によって、性能劣化の抑制を実現できる。

本ツールを実ネットワークへ適用した場合の 1 台当りの性能を見積もるため、実ネットワークを流れるトラヒックの平均パケット長を計測した結果が、図 10 である。同図は、APAN [5] と WIDE [6] を接続するギガビットイーサネットを流れる双方向のトラヒックを 24 時間キャプチャし、1 分ごとの平均パケット長 (図中"+") とトラヒック量 (図中"×") を示している。図の横軸は 2003 年 6 月 17 日から 18 日にかけての時間を、左の縦軸はパケット長 (単位: バイト) を、右の縦軸はトラヒック量 (単位: Mbit/s) を示している。この日 24 時間のパケット長の平均は 861 バイトであった。

このパケット長の値は一例でしかないが、キャプチャ装置 1 台当り 2K - 27K packet/sec のパケット速度では 13M から 180Mbit/s トラヒック量に相当することになる。

5.2 時刻精度と同期

本稿で提案するフロー計測ツールの時刻は、個々のキャプチャ装置の時刻を用いている。本提案では、汎用の分散装置を用いてトラヒックを分散する前提で設計を行ったため、精度の高い時刻を確保するためには、全てのキャプチャ装置に GPS または GPS 程度の精度を持つ時刻を供給する必要がある。通常、ネットワーク機器の設置場所は窓を開放することができないため、GPS アンテナの設置は簡単には出来ない。

一方、キャプチャ装置を多数用いて計測を行う場合、GPS のクロックを分配する他に、分散装置で時刻を取得してパケットを振り分ける際に、パケットにタイムスタンプを付加する方法が有効と考えられる。この場合、本提案の設計に対してキャプチャ装置の変更は小さいため、このような分散装置を開発できれば、キャプチャ装置間の時刻同期を簡略化できる。

5.3 高速化への考察

本提案ではキャプチャ処理に pcap ライブラリを活用したが、Linux で動作する pcap ライブラリでは、内部で recv システムコールが用いられており、パケットが到着する度にカーネルとユーザプログラムのコンテキストが切り替わり、それに付随するオーバヘッドが処理速度に影響与える可能性がある。[7] 更に、本提案では、1 つのパケットのフロー同定処理中に、後続パケットがキャプチャ落ちを起こす恐れがあるため、キャプチャパケットをバッファへ溜める専用のスレッドを装備した。

キャプチャ装置では、主に、フロー同定処理とパケットバッファ処理が動き続けるが、これら 2 つのスレッドの切り替えは、カーネルのプロセススケジューリングに依存しており、パケットバッファ処理とフロー同定処理を適切に平衡に保つことが難しい。キャプチャ装置全体の性能の限界に近い頻度でパケットを受信する場合に、この平衡状態が性能を発揮する鍵となるため、柔軟なプロセススケジューリングが求められる。

処理を高速化する 1 つの案として、カーネルにパケットをバッファさせる方法が考えられる。この手法は、カーネル空間にまとまった数のパケットを保持するメモリを確保し、メモリが一杯になるまで、カーネルが受信パケットのコピーを保存する。メモリが一杯になると、初めてユーザプログラムへパケットをまとめて渡し、ユーザプログラムはフローの同定処理を行う。その間、カーネルはパケット到着の割り込みを受けるたびに、パケットを次のメモリ空間に保存していく。

パケット到着の際の割り込みを有効に活用することで、適切なプロセススケジューリングに近い処理の切り替えが期待でき、キャプチャ性能の改善が見込める。さらに、コンテキスト切り替えのオーバヘッドの影響を抑制できる見込みもある。

6. むすびに

本稿では、トラヒックを分散してリアルタイムにフロー計測を行う分散型リアルタイムフロー計測ツールの要求仕様を明らかにし、仕様を満たすソフトウェアツールの方式を提案した。更に、同ツールを PC-UNIX 上に実装し、性能評価を行い、複数に分散可能なキャプチャ装置について、1 台で最大 27K packet/sec の性能を確認した。また、フロー定義をパターンの連鎖と定義することで、多数のフロー定義を行う状況において、性能の劣化を抑制できることを示した。今後は、高速化に向けたカーネルによるパケットバッファの改良、時刻精度の向上、特定応用向けの最適化、などを進める予定である。

文　献

[1] http://www.mrtg.org/
[2] 佐藤克久、堀合幸次、浅利一善、酒井俐、石川利昭、金子芳久、松田浩："GPS 時刻同期型 NTP サーバーの時刻同期精度について," 東北大学技術研究会報告、192-194, 2001.
[3] J. Quittek, T. Zseby, B. Claise, S. Zander: "Requirements for IP Flow Information Export," Internet Draft (draft-ietf-ipfix-reqs-10.txt), http://www.ietf.org/internet-drafts/draft-ietf-ipfix-reqs-10.txt, June 2003.
[4] G. M. Adelson-Velskii and Y. M. Landis: "An algorithm for the organization of information," Soviet Math. Dokl., 3:1259–1262, 1962.
[5] 北辻佳憲、小林克志、北村泰一、加藤 朗、小西和憲: "APAN 東京 XP の構築と運用," 情報処理学会分散システム/インターネット運用技術研究会. vol.2000-DSM-17, pp.37-42, 2000.
[6] http://www.wide.ad.jp/
[7] Daniel P. Bovet, Marco Cestati: "LINUX カーネル," 株式会社オライリー・ジャパン発行, ISBN4-87311-048-3, 2002.