社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

任意の時間粒度・空間粒度の検索が可能となるフローデータの格納方法の提案

辻　　喜宏† 　大下　裕一† 　村田　正幸†

† 大阪大学大学院情報科学研究科　〒 565–0871 大阪府吹田市山田丘 1–5
E-mail: †{y-tsuji,y-ohsita,murata}@ist.osaka-u.ac.jp

あらまし　近年研究が進められている細粒度の経路制御手法は、入力として経路制御を行う粒度に応じたトラヒック情報が必要である。また、細粒度の経路制御では、時刻により経路制御を行う粒度も異なる。さらに、トラヒック予測と連携した経路制御に関する検討も進められており、トラヒック予測を行うためには、トラヒックの時系列データが必要となる。本稿では、任意の粒度でのトラヒック時系列データを取得可能なフロー情報の観測・格納方法について議論を行う。本稿においては、任意の粒度でのトラヒック時系列データを瞬時に取得できるように、木構造に基づいた、フロー情報格納のためのデータ構造を提案する。その後、そのデータ構造を用いた場合に、情報更新にかかる計算時間、フロー情報の保持に必要な領域の大きさについて議論する。そして、より細粒度のトラヒック情報の格納に向けた課題とその課題を解決する方針について述べる。

キーワード　トラヒック観測、フロー観測、粒度検索、時間粒度・空間粒度、パトリシア木、基数木

Proposal of Storing Flow Data for Retrieval of Arbitrary Spatio-Temporal Granularity

Yoshihiro TSUJI†, Yuichi OHSITA†, and Masayuki MURATA†

† Graduate School of Information Science and Technology, Osaka University 1–5 Yamdaoka, Suita, Osaka, 565–0871, Japan
E-mail: †{y-tsuji,y-ohsita,murata}@ist.osaka-u.ac.jp

Abstract　In this paper, we propose a method to monitor and store the traffic information with any spatio-temporal granularity. In this method, we construct a tree structure for each key of the traffic information. Then, we evaluate our method, considering the calculation time to store and retrieve the traffic information and storage size. In addition, we also discuss the problems and approach for storing traffic information with finer granularity.

Key words　Traffic Monitoring, Flow Monitoring, Granularity Retrieval, Spatio-Temporal, Patricia, Radix Tree

1. はじめに

クラウドコンピューティングに代表されるネットワークサービスが提供されるようになるにつれ、ネットワーク内のトラヒックの時間変動が激しくなっている。ネットワーク管理者は、ネットワーク内のトラヒックを輻輳させることなく収容できるように、ネットワーク内の経路を設定する。しかしながら、経路設計時とトラヒックが大きく変わると、輻輳が発生してしまう。そのため、時間変動の激しいトラヒックを収容するためには、動的に経路を変更する制御が必要となる。

動的に経路を変更する手法として、トラヒックエンジニアリングと呼ばれる手法の検討が進められている [1], [2]。トラヒックエンジニアリングでは、現在のトラヒック状況や将来のトラヒックの予測値を入力とし、入力されたトラヒック状況に適した経路を設定することにより、変化の大きなトラヒックを適切にネットワークに収容する。近年では、OpenFlow [3] のようにフローごとに経路を設定することが可能な技術も確立されつつあり、これらの技術を用いた細粒度のトラヒックエンジニアリングによる環境変動への迅速な対応が期待されている。

トラヒックエンジニアリングを行うためには、その入力となるトラヒック情報を、経路制御を行う制御単位に応じた粒度で随時把握することが必要である。特に、近年検討されている細粒度のトラヒックエンジニアリングでは、複数のフローを集約した集約フロー単位で経路が制御され、また、フローの集約方法もトラヒック状況に応じて変化する。そのため、状況により異なるフローの集約状況に応じた粒度でのトラヒック情報の把

握が必要となる。さらに、安定的なトラヒック収容をめざし、トラヒックエンジニアリングを行う際にトラヒックの予測値を用いることも考えられており [4]、トラヒックの予測を行うためには、制御単位に応じた粒度のトラヒックについて、過去から現在にわたる時系列データが必要となる。

従来、ネットワーク内のトラヒック量の観測は、NetFlow [5]・sFlow [6] や SNMP [7] による MIB 情報の収集によって行われてきた。しかしながら、それらの手法では、固定の粒度の観測しか行うことができない。あらかじめ指定した粒度でトラヒック量の観測を行う手法 [8] も提案されているものの、これらの手法ではトラヒック観測前にトラヒック情報の粒度を指定する必要があり、経路の制御時にトラヒックの観測結果から、制御の要求に応じた任意の粒度でトラヒック時系列データを把握するという要求を満たすことはできていない。

そこで、本稿では、事後に任意の粒度でのトラヒック時系列データを取得可能なフロー情報の観測・格納方法について議論を行う。本稿においては、任意の粒度でのトラヒック時系列データを瞬時に取得できるように、木構造に基づいた、フロー情報格納のためのデータ構造を提案する。その後、そのデータ構造を用いた場合に、情報更新にかかる計算時間、フロー情報の保持にかかるメモリサイズについて議論したのち、それらの問題を解消するための方針について述べる。以降の本稿の構成は次の通りである。まず、2 章で、ネットワークの観測器に対する要件を整理する。そして、3 章で観測器の設計を、フロー情報を格納するデータ構造を中心に述べる。4 章では実データを用いた評価を行い、5 章で 4 章の評価結果をもとにした課題とその課題を解消するための方針について述べる。最後に、6 章では結論と今後の課題についてまとめる。

2. トラヒック観測器に求められる要件

2.1 トラヒック情報取得の要求

本稿では、観測対象のネットワーク機器からポートミラーリング等により、観測対象の全トラヒックをトラヒック観測器が受信できるようにした上で、トラヒック観測を行うものとする。トラヒックエンジニアリングを行う制御サーバなど、トラヒック観測器が観測したトラヒック情報が必要な機器はトラヒック観測器にトラヒック情報取得の要求を送る。

トラヒック情報取得の要求には、取得対象のトラヒック情報の範囲と粒度に関する指定が含まれている。本稿では、トラヒックの範囲は以下の 3 種類のキーにより指定されるものとする。

時間範囲　取得対象のトラヒック情報の開始時刻と終了時刻が指定される。

送信元 IP アドレス　送信元 IP アドレスのプレフィックスが指定される。

宛先 IP アドレス　宛先 IP アドレスのプレフィックスが指定される。

また、トラヒック情報の粒度も同様に以下の 3 種類のキーにより指定されるものとする。

時間範囲　取得対象データ内のタイムスロットの長さが指定される。

送信元 IP アドレス　情報取得粒度における送信元 IP アドレスのプレフィックス長が指定される。

宛先 IP アドレス　情報取得粒度における送信元 IP アドレスのプレフィックス長が指定される。

観測器では、上記の情報を含むトラヒック情報取得要求が到着した場合には、指定された範囲内のトラヒックに関する情報を、指定された粒度ごとにまとめた上で返答を行う。

2.2 トラヒック観測器の要件

2.2.1 情報取得にかかる性能要件

トラヒック観測器では、2.1 節で述べた情報取得要求に対して、指定された範囲のトラヒック情報を指定された粒度で返答することが求められる。その返答は、トラヒック情報を要求したアプリケーションが要求する時間以内に行う必要がある。たとえば、近年研究が進められているデータセンターネットワークを対象としたトラヒックエンジニアリングでは、秒単位での制御が必要とされており [9]、このようなアプリケーションを想定すると、トラヒック情報の返答も遅くとも 1 秒以内に行う必要がある。任意の粒度で情報を返答可能な方法として、時間的に小さい粒度で観測した観測データを各フロー単位で保持しておき、情報取得の要求に合わせて当該データを処理して、情報取得要求で指定された粒度のトラヒック情報を生成するという方法も考えられるものの、この方法では、トラヒック情報取得要求に対して返答を行うまでの時間が長くかかってしまう。そのため、任意のトラヒック情報取得要求に対して瞬時に返答可能なようにトラヒック情報を整理しながら観測することが求められる。

2.2.2 トラヒック観測にかかる要件

a）　データ更新の速度

トラヒック観測器には、ネットワークを流れるパケットが次々に到着し、トラヒック観測器はそれらのパケットの観測処理を行うことが求められる。サンプリングにより、観測対象を減らすことも考えられるが、サンプリングレートが低くなると、サンプリング誤差が大きくなってしまうという問題が生じる。特に、細粒度のトラヒック情報は、サンプリング誤差が大きくなりやすい。そのため、十分な精度で観測するため、パケット到着の速度に合わせたデータの更新速度が求められる。

b）　観測器のリソース

観測器では様々な粒度のトラヒック情報取得要求に対応できるようにトラヒック情報の保存を行う。その結果、保存するトラヒック情報が膨大となってしまうことが考えられる。しかしながら、トラヒック情報が膨大となると、その情報を保存する機器にかけるコストも大きくなってしまうという問題が生じる。そこで、少ないメモリやディスクで、トラヒック観測器で観測されたトラヒック情報を保存可能であることが望ましい。

3. トラヒック観測器の設計

本稿のトラヒック観測器では、指定された最細粒度の時間粒度（t 秒）を 1 タイムスロットとし、タイムスロット単位でトラヒックの観測を行うものとする。トラヒック観測器では、(1) 現在のタイムスロットのトラヒック情報、(2) 過去のタイムス

ロットのトラヒック情報の 2 種類のトラヒック情報を保持する。以下に、それぞれの情報の保持を行う機構について述べる。

3.1 現在のタイムスロットのトラヒック情報

現在のタイムスロットのトラヒック情報は、送信元 IP アドレス・宛先 IP アドレスの組で定義されるフローと、そのフローに対応するカウンタを用いて保持される。パケット到着時には、そのパケットが属するフローに該当するカウンタの値を更新する。

現在のタイムスロットのトラヒック情報は、頻繁に参照・更新されるため、高速な読み書きが可能な SRAM 上に保存する。そして、タイムスロット終了時に、当該タイムスロットの情報は、すべて過去のタイムスロットトラヒック情報としてエクスポートされ、SRAM 上の該当データはすべてリセットされる。

3.2 過去のタイムスロットのトラヒック情報

終了したタイムスロットのトラヒック情報は、すべて過去のタイムスロットのトラヒック情報として保存される。トラヒック情報取得要求がトラヒック観測器に到着した際には、該当するトラヒック情報を、過去のタイムスロットのトラヒック情報から検索し、叙法取得要求で指定された粒度に合わせて返答する。そのため、過去のタイムスロットのトラヒック情報は、情報取得要求に合わせて瞬時に必要な情報が取り出せるように、整理されたものである必要がある。

本稿では、任意の粒度で集約したトラヒック情報を参照可能なデータ構造として、木構造をもとにしたデータ構造を提案する。提案するデータ構造を図 2 に示す。図に示されるように、提案するデータ構造では、時間、送信元 IP アドレス、宛先 IP アドレスの、各粒度に該当する 2 分木を構築する。各木構造では、葉ノードが最細粒度の情報を持ち、子ノードの情報を集約した情報を親ノードが持つ。また、時間粒度に該当する木の各ノードは、そのノードが管理する時間において到着したトラヒックに関する送信元 IP アドレス別の情報を保持する木の根へのポインタを持つ。さらに、送信元 IP アドレス別の情報を保持する木の各ノードは、当該木が管理している時間において、当該送信元 IP アドレス宛のトラヒックを受け取った宛先 IP アドレスごとのトラヒック情報を管理する木へのポインタを持つ。そして、宛先 IP アドレスごとのトラヒック情報を管理する木のみが、トラヒック情報（パケット数、トラヒックサイズ）を持つ。

この構造では、指定された粒度にしたがって、木を探索することにより、任意の粒度のトラヒック情報を取得することが可能となる。つまり、時間粒度、送信元 IP アドレスのプレフィックス長、宛先 IP アドレスのプレフィックス長が指定されると、まず、時間粒度の木を探索し、指定された時間粒度に該当するノードに到達する。その後、到達した時間粒度の木のノードが保持するポインタをたどり、送信元 IP アドレスごとの情報を管理する木に到達する。送信元 IP アドレスごとの情報を管理する木においても、同様に、指定された送信元 IP アドレスの範囲内で情報取得粒度のプレフィックス長に該当するノードに到達する。そして、そのノードが保持するポインタをたどり、宛先 IP アドレスごとの情報を管理する木に到達する。最後に宛先 IP アドレスごとの情報を管理する木においても、同様に、指定された宛先 IP アドレスの範囲内で、指定された情報取得粒度のプレフィックス長に該当するノードに到達し、そのノードが保持するカウンタ値を得る。このように、再帰的に木をたどることにより、任意の粒度のトラヒック情報を得ることが可能となる。

過去のタイムスロットのトラヒック情報の更新は、タイムスロット単位でしか発生しない。そのため、頻繁な更新は発生しないため、DRAM などの SRAM よりもより大容量のメモリや、SSD・HDD といった外部記憶装置に保存可能である。

以降、過去のタイムスロットのトラヒック情報の保持するためのデータ構造において、各木の構成方法、データ構造における操作について述べる。

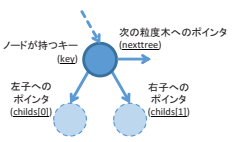

図 1　木構造の各ノード

図 2　観測サーバが持つデータ構造

3.2.1 各木の構成方法

a） 時間粒度の情報管理を行う木

時間粒度の木構造は、N 分木で表現するものとし、葉ノードが観測タイムスロット (t 秒) ごとの粒度に該当し、親ノードは、各子ノードの全タイムスロットを包含した、より大きな時間粒度のデータに該当するものとする。以降の本稿の評価・考察では、時間粒度の情報を管理する木は、図 2 のように 2 分木で表されるものとする。

b） 空間粒度（IP アドレス）

フロー情報の IP アドレスとして、(32 から 0 まで範囲での) プレフィックス長を含む IP アドレスを想定しているので、IP アドレスの粒度同士は包含関係となりうるため、時間粒度の木構造と同じく、図 2 のように 2 分木の形で階層的に管理される。階層の深さは IP アドレスのビット長に相当し 32 である。各ノードは一つの IP アドレスの空間粒度に対応し、それが持つ情報は以降の空間粒度キーのタップルで管理される木構造へのポインタである。

c） 送信元・宛先 IP アドレスごとの情報管理を行う木

送信元・宛先の IP アドレスの情報は、IPv4 を想定すると、各 32bit であらわされる。そのため、深さ 32 の完全 2 分木によりすべての IP アドレスに対応する情報を保持することができる。この完全 2 分木において、親ノードは、子ノードに対応するトラヒックの集約情報を保持すると、プレフィックス長 L のトラヒック情報は、深さ L の地点のノードが持っており、葉

ノードまでたどることなく、任意の粒度のトラヒック情報を取得することが可能となる。

トラヒック観測器に到着するフローの IP アドレスには偏りがあると考えられる。そのため、完全 2 分木の構造で、トラヒック観測データを保存した場合、該当するトラヒック情報が存在しないノードが多数存在するようになる。そのため、到着したフローの IP アドレスに対応しつつ、不要なノードを作らないような木構造を採用することにより、トラヒック情報保存にかかるデータのサイズを縮小することが可能となる。

本稿では、必要な情報を保持しつつ、トラヒック情報保存にかかるデータのサイズを縮小させるため、送信元・宛先 IP アドレスに対応する情報を保存する各木は、完全 2 分木ではなく、バイナリ列の集合を効率よく管理することができるデータ構造である、パトリシア木を用いる [10], [11]。パトリシア木は、送信元・宛先 IP アドレスに対応する各木をパトリシア木とすることにより、完全 2 分木を用いた場合と同様に、任意の粒度の集約情報を瞬時に探索することができ、かつ、木に不要なノードを削減することにより、トラヒック情報保存にかかるデータサイズを縮小できる。

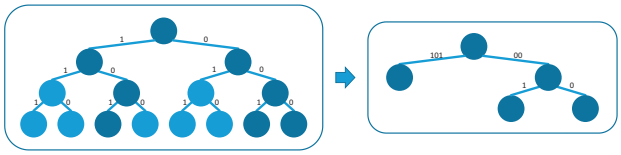

図 3　完全二分木と対応するパトリシア木の例

3.2.2　データ構造に対する処理

a）参　　照

取得対象の粒度と範囲に関する情報を含むトラヒック情報取得要求が到着すると、トラヒック観測器では、その要求に合わせたトラヒック情報を、過去のタイムスロットのトラヒック情報から検索し、返答する。この検索は、上述のように、各粒度に対応する情報を保持する木をたどることにより可能となる。その際に、各木において、取得対象の粒度の情報を保持するノードは、指定された粒度よりも細かい粒度の情報を持つ子が存在するノードであり、木の根からたどることにより発見可能である。

b）更　　新

過去のタイムスロットのトラヒック情報は、タイムスロットごとに更新が発生する。タイムスロット終了時に、現在のタイムスロットの情報を保持するメモリ領域からデータが転送され、転送されたデータを木構造に整理を行う。新たなタイムスロットのトラヒックデータ到達した際には、時間粒度の情報を管理する木に対して、当該タイムスロットのデータに該当するノードを葉として追加した上で、葉から根までたどりながら途中のノードに該当するデータの更新を行う。

4.　評　　価

本節では、提案するトラヒック観測器をプログラムとして実装し、実際のトラヒックキャプチャデータを処理させることにより、その性能を評価する。

4.1　評 価 環 境

4.1.1　実 装 環 境

本評価では、トラヒック観測器は、libpcap ライブラリを用いて実装し、GCC でコンパイルを行った。そして、Intel(R) Xeon(R) CPU、32768MB のメモリを搭載した計算機上で動作させた。

4.1.2　トラヒックデータ

本稿では、2014 年 2 月 7 日に大阪大学のゲートウェイを通過したパケットヘッダをキャプチャしたデータを用いた。本キャプチャデータは、pcap 形式で保存される。IP アドレスの下位 16 ビットをハッシュにかけることにより匿名化の処理が行われている。本匿名化により、当該キャプチャデータに含まれている IP アドレスは、実際の通信に用いられた IP アドレスとは異なるものとなっているものの、(1) 上位 16bit は実際の通信に用いられた IP アドレスであり、(2) 匿名化の処理がされているものの、同一フローの IP アドレスは同一となるように処理されている。そのため、この匿名化は、以降の評価に対して影響を与えない。

4.1.3　トラヒック観測器のパラメータ設定

トラヒック観測器は、タイムスロットごとにトラヒック情報のデータを過去のタイムスロットの情報として保存する。このタイムスロットの長さ t は、以下の影響を持つ。(1) 保存される累計トラヒック情報量に影響を与える：タイムスロットが短く一つのフローが複数のタイムスロットにわたる場合は、当該フローの情報が複数のタイムスロットで保存されるため、保存される累計トラヒックエントリ数が増加する、(2) 時間粒度の木の深さが深くなる。そこで、本稿では、0.1 秒, 1 秒, 10 秒の三通りのタイムスロットを用いた場合について、比較評価する。

4.2　観測フロー数

観測器の評価を行う前に、本評価に用いたデータの性質について述べる。図 4 は、本キャプチャデータをもとに、各タイムスロット内で観測されるフロー数の時間変化を示したものである。タイムスロットの長さをどのように設定したとしても、本キャプチャデータでは、12:00–13:00 のフロー数が多いことが分かったので、本評価では、提案するトラヒック観測器が、フロー数が多い時間帯に対応できることを示すため、フロー数が多い 12:00–13:00 のデータを評価に用いる。

また、各タイムスロットで観測されたフローを観測期間内で合計した値を表 1 に示す。以降、この値を累計フロー数と呼ぶ。累計フロー数を数える際には、複数のタイムスロットにまたがって流れているフローは、タイムスロットごとに別のフローとしてカウントする。累計フロー数はトラヒック情報を保存するサイズに大きな影響を与え、累計フロー数が大きくなれば、そのトラヒック情報を保存するのに必要な領域のサイズも大きくなると考えられる。

表に示されるように、タイムスロットの長さが長くなるほど、1 時間で観測された累計フロー数は少なくなることが分かる。

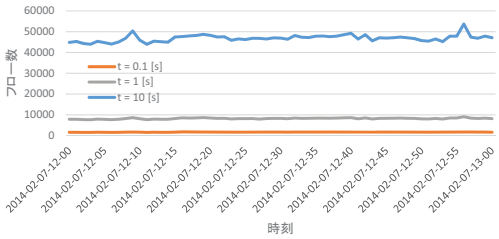

図 4　各タイムスロットで観測されるフロー数の変化

表 1　各タイムスロットに対する観測の累計フロー数

観測期間	タイムスロットの長さ		
	0.1 秒	1 秒	10 秒
2014/02/07 12:00–13:00	60791996	29690207	16918474

表 2　累計観測フロー 1 つあたりの木構造内ノード数

タイムスロットの長さ	0.1 秒	1 秒	10 秒
ポインタを持つノード	76.0	72.2	55.7
カウンタを持つノード	90.8	86.4	66.1

4.3　評価結果

4.3.1　トラヒック情報のサイズ

本稿では、トラヒック情報を格納するのに必要な領域の大きさを見積もるため、まず、各タイムスロットの長さごとに、観測された累計フロー数とそのフロー数の情報を格納する際に構築される木の大きさについて議論する。その後、その木構造の大きさをもとに、トラヒック情報を保存するのに必要なサイズの大きさを見積もる。

a）　累計フロー数と木構造内のノード数の関係

本稿で過去のタイムスロットのトラヒック情報を保存するために用いる木構造では、(1) 時間粒度の木や送信元 IP アドレスの木のノードのように、次の木へのポインタのみを持ち、トラヒックカウンタを持たないノード、(2) 宛先 IP アドレスの木のノードのように、トラヒックカウンタを持つノードの 2 種類のノードがある。そこで、本節では、大阪大学のゲートウェイに到着した、20 万フローに関する情報を過去のタイムスロットのトラヒック情報を保存する木構造に保存し、観測された累計フロー数に対して必要となる木構造の中の 2 種類のノードの数を調べた。

表 2 に結果を示す。表より、タイムスロットが長くなるにつれて、1 フローあたりの木構造内のノード数は減ることが分かる。これはタイムスロットが長くなると、一つのタイムスロットで扱うフロー数が多くなってしまうことによるものである。同一のタイムスロットで異なるフローを表現するときは時間粒度の木構造に分岐は生じず、以降の送信元 IP アドレスの木や宛先 IP アドレスの木の分岐によって表現されることになり、異なるタイムスロット間で異なるフローを表現する三つの木構造で表現する場合に比べてノード数は少なくなると考えられる。$t = 0.1, 1.0, 10$ の三つの場合でフロー数の違いは図 4 で確認することができるが、その図で示されるほどフロー集合表現のためのノード数に違いが出ない理由としては、フロー寿命に対してタイムスロットの長さが短い場合、異なるタイムスロットに分布することになる。そのようなケースはタイムスロットの長さが短い場合に起きやすく、そのようなケースのときにはより粗い時間粒度でみたとき一つのフローとして認識されるため追加されたフロー数に対してノード数が少しは抑えられることになる。前者の影響の方が強く、結果として表 2 のようにタイムスロットの長さが長い場合の木構造のノード数はタイムスロットの長さが短い場合に比べて少し小さくなったと考えられる。

b）　トラヒック情報のサイズの見積り

表 2 をもとに、トラヒック情報の保存に必要な領域の大きさを見積もる。ポインタ情報のみ持つノードは、自身の子ノードへのポインタ 2 つと、次に探索される情報のための木の根へのポインタの合計 3 種類のポインタを持つ。それに対して、トラヒックカウンタを持つノードは、自身の子ノードへのポインタ 2 つとトラヒックカウンタを持つ。本見積りでは、ポインタに 8Byte、トラヒックカウンタに 8Byte の領域が必要であるとした。

表 3　1 時間分のトラヒック情報保存に必要な領域のサイズ (GByte)

タイムスロットの長さ	0.1 秒	1 秒	10 秒
必要な領域のサイズ	243.4	113.0	49.5

表 1 の情報と合わせ、1 時間分のトラヒック情報を格納した際のトラヒック情報のサイズを表 3 に示す。表より、タイムスロットが 10 秒であれば、50GByte 程度の情報で大阪大学のゲートウェイにおいて 1 時間で観測された全情報を保持できることが分かる。しかしながら、タイムスロットが 0.1 秒と短くなると、情報の保持に必要な領域のサイズが著しく大きくなる。そのため、時間的に極細粒度のトラヒック情報を保持するためには、さらに領域を削減するための工夫が必要となる。

4.3.2　フロー情報追加にかかる時間

各タイムスロットで観測されたトラヒック情報は、タイムスロットが終わると、過去のタイムスロットのトラヒック情報を保存する木構造に追加される。この処理は、タイムスロット内で観測された各フローの情報を 1 つずつ木構造に追加することにより行われる。そこで、本評価では、各フローの情報を木構造に格納するのにかかる時間を計測した。その結果、タイムスロットの長さが 0.1, 1, 10 秒の場合、それぞれ 0.47, 0.36, 0.27 ミリ秒の計算時間を要した。タイムスロットが短ければ同じ数のフローを収納するために必要なノード数は多く、ノード数が多いとフロー情報追加時に更新するノードが増えてしまい計算量が大きくなってしまう。図 4 より、各タイムスロットあたりの観測されるフロー数は多い時間帯では、タイムスロットの長さが 0.1 秒であれば 4344、タイムスロットの長さが 1 秒であれば 26117、タイムスロットの長さが 10 秒であれば 76265 である。そのため、各タイムスロットのデータを木構造に追加するのにかかる時間は、タイムスロットの長さが 0.1 秒であれば 2 秒、タイムスロットの長さが 1 秒であれば 9.4 秒、タイムスロットの長さが 10 秒であれば 20.6 秒となる。リアルタイム観測を行うためには、オーダー一個分だけ計算量を下げる必要がある。

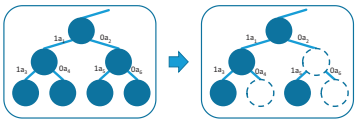

図 5　情報を保持するノードの削減

4.3.3　データ構造からフロー情報取得にかかる時間

最後に、木構造に保存されたデータから、指定されたトラヒック情報を取得するのにかかる時間を計測した。その結果を表 4 に示す。表中の値は順にタイムスロットの長さとして 0.1 秒, 1 秒, 10 秒を持つ、送信元 IP と宛先 IP に関して、それぞれ指定ビット長が 0, 15, 31 であるクエリを与えて時間を計測した。なお時間粒度はランダムに指定し、それぞれ 5 万だけ回検索の関数を回したときの平均時間である。表より、指定された粒度によらず、短時間で必要なトラヒック情報を取得することができていることが分かる。

表 4　検索粒度の粒度を変更した時の変化させたときの計算時間 [us] の変化（タイムスロットの長さ 0.1 秒, 1 秒, 10 秒）

		宛先 IP の指定ビット長		
		0	15	31
送信元 IP の指定ビット長	0	(0.2, 0.0, 0.2)	(0.2, 0.0, 0.2)	(0.2, 0.2, 0.2)
	15	(0.4, 0.2, 0.0)	(0.2, 0.2, 0.2)	(0.2, 0.2, 0.0)
	31	(0.2, 0.2, 0.2)	(0.0, 0.2, 0.2)	(0.2, 0.2, 0.2)

5.　提案手法の問題と改善に向けた方針

4. 章の結果より、本稿で提案したトラヒック観測器では、任意の粒度でのトラヒック情報を短時間で検索することができることが分かった。その一方、細粒度のトラヒック情報を保存すると、保存に必要な領域のサイズが著しく大きくなることが分かった。また、本稿では、フローは送信元 IP アドレス・宛先 IP アドレスの組み合わせにより定義されるものとしたが、通信に用いられているポート番号も含め、（送信元 IP アドレス・宛先 IP アドレス・送信元ポート番号・宛先ポート番号）の組み合わせにより、より細かな粒度で観測対象のフローを定義することも考えられる。その場合にも、本稿の提案手法では、ポート番号をキーとして管理する木構造を追加することにより適用可能であるが、保持が必要な領域のサイズがさらに大きくなると考えられる。

必要な領域のサイズを削減する手法として、木構造のうち、情報を持つノード数を削減することが考えられる。たとえば、図 5 のように、各木構造において、同一の親をもつ 2 つのノードのうち、一方のみデータ（次のキーに対応する木構造へのポインタやトラヒックカウンタ）を持たせることにより、データの保持が必要なノード数を半減することができる。しかもこの場合であっても、データを持たないノードに該当するトラヒック情報は 親ノードの情報 − データを持つ子ノードの情報 の計算を行うことにより可能であると考えられる。しかしながら、このアプローチを用いた場合は、トラヒック情報の格納に必要な領域のサイズを削減することができるものの、必要な情報を得るために上述の演算をする必要が生じる。そのため、このアプローチの有用性は、情報取得のための演算にかかる時間も含めて議論する必要があり、今後の課題である。

6.　まとめと今後の課題

本稿では、任意の粒度でのトラヒック時系列データを取得可能なフロー情報の観測・格納方法について議論を行った。本稿においては、任意の粒度でのトラヒック時系列データを瞬時に取得できるように、木構造に基づいた、フロー情報格納のためのデータ構造を提案した。その後、そのデータ構造を用いた場合に、情報更新にかかる計算時間、フロー情報の保持に必要な領域の大きさについて議論した。

今後の課題としては、トラヒック情報を保持するのに必要な領域のサイズを削減する方法に関する議論が挙げられる。

謝　　辞

本研究の一部は、文部科学省科学研究費補助金若手研究 (B)23700077 によっている。

文　　献

[1] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for Traffic Engineering Over MPLS," RFC 2702, Sep. 1999.

[2] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "MATE: MPLS Adaptive Traffic Engineering," in *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3. IEEE, 2001, pp. 1300–1309.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[4] O. Tatsuya, O. Yuichi, M. Masayuki, T. Yousuke, I. Keisuke, and S. Kohei, "Traffic Prediction for Dynamic Traffic Engineering Considering Traffic Variation," in *Proceedings of IEEE Globecom*, Dec. 2013, pp. 1592–1598.

[5] B. Claise, "Cisco Systems NetFlow Services Export Version 9," RFC 3954, Oct. 2004.

[6] P. Phaal, S. Panchen, and N. McKee, "InMon Corporation ' s sFlow: A Method for Monitoring Traffic in Switched and Routed Networks," RFC 3176, 2001.

[7] J. Case, M. Fedor, M. Schoffstall, and C. Davin, "A Simple Network Management Protocol (SNMP)," RFC 1098, Apr. 1989.

[8] L. Yuan, C.-N. Chuah, and P. Mohapatra, "ProgME: Towards Programmable Network MEasurement," *IEEE/ACM Trans. Netw.*, vol. 19, no. 1, pp. 115–128, Feb. 2011.

[9] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: Fine Grained Traffic Engineering for Data Centers," in *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*, ser. CoNEXT '11. New York, NY, USA: ACM, 2011, pp. 8:1–8:12.

[10] W. Doeringer, G. Karjoth, and M. Nassehi, "Routing on Longest-Matching Prefixes," *IEEE/ACM Transactions on Networking (TON)*, vol. 4, no. 1, pp. 86–97, 1996.

[11] D. E. Knuth, "The Art of Computer Programming. vol. 3: Sorting and Searching," *Atmospheric Chemistry & Physics*, vol. 1, 1973.