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

2006－ＤＳＭ－41（８）
2006／5／12

次世代バックボーン向け大容量トラフィック・コレクタの提案

小林 淳史* 松原 大典** 木村 真吾*** 齋藤 元之***
廣川 裕* 坂本仁明* 石橋 圭介* 山本 公洋*

*日本電信電話株式会社 情報流通プラットフォーム研究所 〒180-8585 東京都武蔵野市緑町 3-9-11
**株式会社日立製作所 中央研究所 〒185-8601 東京都国分寺市東恋ヶ窪 1-280
***NTT アドバンステクノロジ株式会社 〒180-8585 東京都武蔵野市緑町 3-9-11
E-mail: * akoba@nttv6.net, (hirokawa.yutaka, sakamoto.hitoaki, ishibashi.keisuke, yamamoto.kimihiro)@lab.ntt.co.jp
**d-matuba@crl.hitachi.co.jp ***saitou@nttv6.net, shingo.kimura@ntt-at.co.jp

あらまし 膨大なトラフィック情報を受信する次世代バックボーン向けのトラフィック・コレクタは、高速化・大容量化が必要となる．本研究では、近年益々拡大するネットワーク規模に対応して、柔軟に拡張可能なトラフィック・コレクタの構成モデルの提案する．ルータから配信されるフロー情報を受信し、集約・蓄積後、更に上位ノードに配信するフロー・コンセントレータとよぶトラフィック情報集約装置を用いて、トラフィック・コレクタの大容量化、拡張性の実現を目指した．また、コンセントレータのプロトタイプ開発をもとに、その効果を評価している．

キーワード IPFIX, NetFlow, コンセントレータ, フロー

A Proposal of Traffic Collector for Next Generation Backbone

Atsushi KOBAYASHI* Daisuke MATSUBARA** Shingo KIMURA*** Motoyuki SAITOU***
Yutaka HIROKAWA* Hitoaki SAKAMOTO* Keisuke ISHIBASHI* Kimihiro YAMAMOTZO*

*NTT Information Sharing Platform Laboratories 3-9-11 Midori-cho, Musashino, Tokyo, 180-8585 Japan
**Hitachi, Ltd., Central Research Laboratory 1-280 Higashi-Koigakubo, Kokubunji, Tokyo, 185-8601 Japan
***NTT Advanced Technology Corporation 3-9-11 Midori-cho, Musashino, Tokyo, 180-8585 Japan
E-mail:* akoba@nttv6.net, (hirokawa.yutaka, sakamoto.hitoaki, ishibashi.keisuke, yamamoto.kimihiro)@lab.ntt.co.jp
** d-matuba@crl.hitachi.co.jp *** saitou@nttv6.net, shingo.kimura@ntt-at.co.jp

Abstract In large-scale networks, a single central collecting process might not be able to process all flow records exported. A way to increase scalability is sharing the load of processing flow records with a set of Flow concentrators that aggregate flow records received from routers and export the aggregated records.

An Flow concentrator is located between one or more routers and one or more Traffic Collectors. Flow concentrator acts as collector towards exporting processes sending flow records and it acts as exporter towards collectors that receive flow records exported by the concentrator. This dual-role architecture allows cascading concentrators and adjusting the number of Flow concentrators to the size of a given network.

Keyword IPFIX, NetFlow, concentrator, flow

1. はじめに

近年、ネットワークの設備計画や DDoS 攻撃などの異常トラフィックを検知する目的から、ネットワーク上に流れているトラフィックをモニタする機能（sFlow,NetFlow）に注目が集まっている．しかし、ネットワーク上に流れるトラフィック量は、年々増加傾向にあり、トラフィック測定をする際に、出力されるフロー情報も膨大となりつつある．現在、日本のインターネットトラフィックは、年２倍のペースで増え続けており、ブロードバンド加入者のトラフィック量は、230Gbps 程度との報告がある[1].

これを 5 年後の主要 ISP の PoP 単位のトラフィックに換算すると約 100Gbps 相当になると推定され、サンプリングレート 1/1024 にてパケットサンプリングを行ったとしても、250K レコード/分のフロー情報がトラフィック・コレクタに配信されることとなる．

膨大なフロー情報は、ルータで事前に集約することによって情報量を削減できる一方、集約によって詳細なトラフィック情報が観測できないという問題が発生する．このため、膨大なフロー情報を効率的に集約・蓄積する手法が必要とされている．

本稿では、ルータとトラフィック・コレクタ間に、主に蓄積・集約機能を実現するフロー・コンセントレータと呼ぶノードを配置し、そのノードを中心としたフロー情報の配信・集約・蓄積を効率的に実現するアーキテクチャを提案する．2 節では、フロー・コンセントレータの概要を示し、3 節ではフロー・コンセントレータの内部構成及び外部接続構成を提案する．4 節では、フロー・コンセントレータの実装に向けた考慮点を示し、5 節にてプロトタイプ開発による評価を示す．

2. フロー・コンセントレータ概要

フロー情報の配信プロトコルについては、sFlow,NetFlow 等様々なプロトコルが使用されてきた．近年、NetFlow ver.9 が開発され[2]、フロー情報に含まれる情報要素をプロトコルのフォーマットに依存することなく、任意に変更することが可能となった．これは、フロー情報のフォーマットを示すテンプレートを配布することで実現されており、情報要素の追加・変更を柔軟に行うことが可能となっている．現在、IETF にて議論されている新たなフロー情報配信プロトコルである IPFIX もこのテンプレート方式を採用しており[3]、このテンプレート方式に対応したルータ機器の実装方式について議論されている[4]．

フロー・コンセントレータは、IPFIX においてフロー情報を集約するノードとして位置づけられており[5]、集約手法については柔軟な手法が提案されている[6]．図 1 に IPFIX にて定義されているノードを示す．

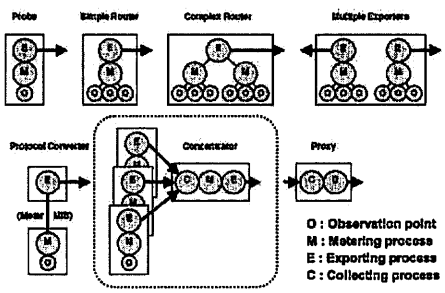

図 1. IPFIX プロトコルに関連するノード群．

フロー・コンセントレータは、Collection process, Metering process, Exporting process をもつノードとして位置づけられているが、詳細な内部プロセス構成や外部接続に関するリファレンス・モデルについては、規定されていない．本稿では、このフロー・コンセントレータに着目し、大規模ネットワークのフロー情報収集に必要とされる集約・選択・蓄積機能に関して、以下を実現するためのノードと位置づける．

- 配信先のトラフィック・コレクタに特別な機能を実装せず、蓄積・集約処理双方の負荷分散を実現する．
- 分散配置されたフロー・コンセントレータ内のフロー情報の検索を単一トラフィック・コレクタから可能とする．
- ネットワークの規模に応じて、コンセントレータの台数を変えることにより、スケーラブルにフロー情報の蓄積・集約することを可能とする．
- 集約前のトラフィックデータを蓄積し、詳細なフローデータの参照を可能とする．

上記の点を踏まえて、次節では、フロー・コンセントレータの構成モデルについて提案する．

3. フロー・コンセントレータ構成モデル

3.1. 外部接続構成

図 2 にフロー・コンセントレータの外部接続構成モデルを示す．フロー・コンセントレータは、NetFlow ver.9 及び IPFIX プロトコルを用いて、ルータとトラフィック・コレクタ間に介在するノードである．また、フロー・コンセントレータを多段に接続して、集約レベルを段階的に上げていくことを可能とする．

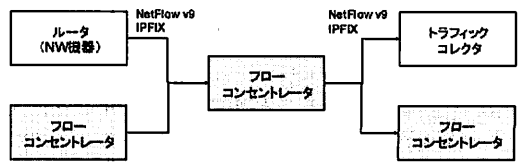

図 2.フロー・コンセントレータの外部接続構成モデル．

3.2. 内部プロセス構成

内部のプロセス構造については、テンプレート方式をもつフロー情報配信プロトコルに対応して、柔軟なプロセス・モデルを検討した．主な機能プロセスは、フロー情報をポリシーに従って取捨選択するプロセスと、フロー情報内の情報要素を任意に選択し集約するプロセスから構成される．

これにより、任意のフロー情報の抽出を可能とし、任意の情報要素をキー情報とする集約を可能とする．例えば、特定の INPUT-IF をもつフロー情報のみを選択し、送信先 IP アドレスの Prefix レンジによる集約や BGP Next-Hop アドレスによる集約を実現可能とする．

また、集約によって失われる詳細な情報を保持するため、集約の前段でルータから配信されるフロー情報をコンセントレータ内に蓄積可能とする．これにより、上位ノードから詳細な情報を参照する場合に利用可能な構成とした[7]．図 3 に内部プロセス構成モデルを示す．

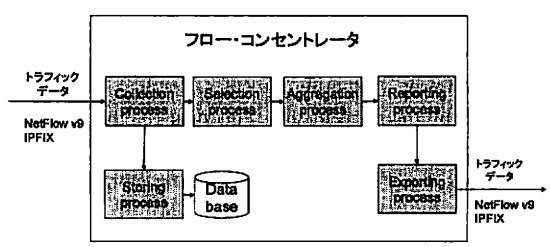

図 3. フロー・コンセントレータの内部構成モデル．

フロー・コンセントレータは 6 つのプロセスから構成される．各プロセスの概要と処理イメージを以下に示す．

(1) 収集プロセス(Collection process)
ルータもしくは他のコンセントレータから配信されるフロー情報を受信する機能をもつ．本プロセスは、セッション単位に生成され、送信元 IP アドレス,ポート番号の情報を管理する．また、フロー情報内に含まれる情報要素と対応するデータ形式を管理する．収集プロセスで受信したトラフィックデータは、複数の選択プロセスもしくは蓄積プロセスに引き継ぐことを可能とする．

(2) 蓄積プロセス(Storing process)
受信したフロー情報をデータベースに蓄積する機能をもつ．この際、蓄積プロセスは、情報要素単位にデータベースへ蓄積するかどうかを判定する．これらの情報は、予めユーザにより設定され、指定された情報要素のみを含むフロー情報をデータベースに格納する．

(3) 選択プロセス(Selection process)
配信されたフロー情報を集約プロセスに引き継ぐか否かを判定する．これらの条件は、予めユーザにより設定される．判定においては、各情報要素単位に条件文を指定し、全ての条件に合致したフロー情報のみを次のプロセスに引き継ぐ．

(4) 集約プロセス(Aggregation process)
送信元 IP アドレスなどの値をキー情報として、ある時間内のトラフィックデータを集約する機能をもつ．その際に、Byte 数などのカウンタ情報については合算し、集約フロー情報を生成する．フロー情報内のどの情報要素をキー情報とするか、カウンタ情報にするかは、予めユーザにより指定される．集約したフロー情報は、次の配信プロセスに引き継がれる．この他、このプロセスでは、集約により失われてしまう情報の補填も行う．例えば、集約したフロー数やフロー当たりの平均・最小・最大 Active 時間などを生成し、集約フロー情報に追加することによって、集約時に失われるフロー毎の情報を補填する．

(5) レポート/配信プロセス (Reporting/Exporting process)
レポートプロセスでは、配信するテンプレートを管理し、配信プロセスでは、トラフィック・コレクタもしくは他のコンセントレータへトラフィック情報を配信する機能をもつ．

上記の内部プロセス・モデルで示すフロー・コンセントレータを階層的に接続することにより、ルータから配信される詳細なフロー情報を格納しつつ、段階的に集約を行うことを可能とする．また、市販のコレクタと接続することで、コレクタのアプリケーションに合わせてフロー情報を振り分けることを可能とする．例えば、トラフィック交流を集計するコレクタには、集約効率の高い集約手法を用いて一元的にトラフィック情報を集め、DDoS 検知などの多様な分析を行うコレクタには、重要顧客ユーザ向けのトラフィック情報のみを選択して配信することが可能となる．大規模ネットワークでの利用例としては、PoP 単位にコンセントレータを配置し、トラフィック情報を効率的に蓄積・配信することで、設備管理ネットワークへの負荷を抑えることも可能となる．図 4,5 に概念図を示す．

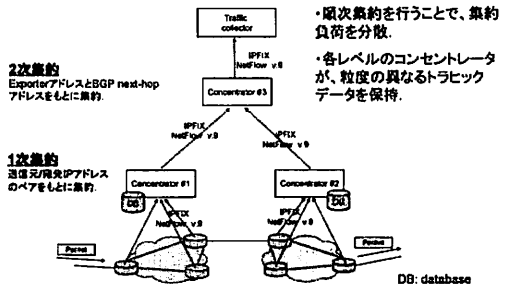

図 4. フロー・コンセントレータの階層接続モデル．

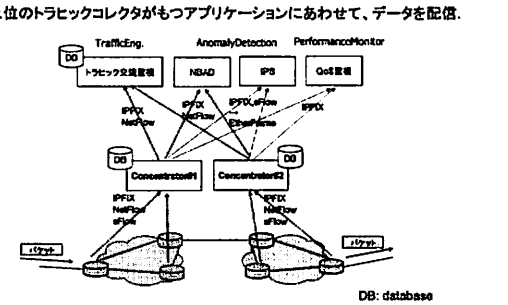

図 5. フロー・コンセントレータの分散配信モデル．

4. フロー・コンセントレータの検討課題

4.1. 柔軟な集約機能の検討

3.2節で述べた各プロセスは、予めユーザが設定したパラメータをもとに動作する．これらの指示パラメータは、情報要素単位に設定すべきものがあり、何種類かの動作指示語を定義することで、各プロセス単位の動作指示テンプレートを作成することができる．

例えば、蓄積プロセスで使用する動作指示テンプレートについては、動作指示語として、"store", "discard"を指定する．各情報要素単位に、"store", "discard"のどちらかを指定することで、"store"であるものは、データベースに格納し、"discard"であるものは、当該情報要素を破棄する．また、集約プロセスにおいては、動作指示語として、"key", "keep", "discard"を指定する．"key"に指定された情報要素は、集約後のフロー情報のキーとされ、"keep"と指定された情報要素は、パケット数などであれば、合算等の処理がされる．以下に、動作指示テンプレートの例を示す．

蓄積指示テンプレート

ID	情報要素名	動作指示語
01	送信元IPアドレス	"store"
02	送信元ポート	"store"
03	宛先IPアドレス	"store"
04	宛先ポート	"store"
05	BYTE数	"store"
06	PACKET数	"store"
0>	プロトコル	"store"
08	INPUT IF INDEX	"discard"
09	OUTPUT IF INDEX	"discard"
20	送信元プレフィックス値	"discard"
21	宛先プレフィックス値	"discard"

集約指示テンプレート

ID	情報要素名	動作指示語
01	送信元IPアドレス	"key"
02	送信元ポート	"discard"
03	宛先IPアドレス	"key"
04	宛先ポート	"discard"
05	BYTE数	"keep"
06	PACKET数	"keep"
0>	プロトコル	"discard"
08	INPUT IF INDEX	"discard"
09	OUTPUT IF INDEX	"discard"
20	送信元プレフィックス値	"key"
21	宛先プレフィックス値	"key"

図 6. 指示テンプレート（例）．

接続した複数のコンセントレータが連携して動作するために、蓄積,選択,集約の各プロセスに必要なパラメータを外部ノードから設定し、参照することが必要とされる.

設定方式としては、従来型の SNMP を用いる方式と現在、IETF-WG にて検討中の NETCONF を用いる方式がある[8]. 集約機能の実装については、将来的に NW 機器(ルータ等)に実装される可能性もあることから、処理負荷の少ない SNMP 方式とした.

コンセントレータの MIB オブジェクトについては、各プロセス毎に必要なパラメータを定義し、各プロセスのインスタンスを連携させる構成とした. これは、IETF-WG にて検討している PSAMP-MIB の構成を参考にしている[9]. コンセントレータ MIB の特徴は、収集プロセスにて管理される collectorMIB オブジェクトとコンセントレータ機能（選択・集約）に特化した concentratorMIB オブジェクトの 2 つのモジュールをもとに構成される[10]. collectorMIB オブジェクトは、一般的なコレクタにおいても利用可能である.

将来的には、構造化が容易であり、柔軟性のある XML をもとにした NETCONF の方式についても検討する必要がある.

4.2. 蓄積方式の検討

一般的にトラフィック・コレクタで使用されている蓄積方式は、主に検索の利便性から RDBMS が使用されている. しかし、一旦蓄積したトラフィック情報の情報要素に対する更新処理は発生しないため、大容量化・高速化の観点から RDBMS が最適であるとは限らず、より高速な蓄積方式について検討が必要である.

また、近年、オープンソースで作成されるトラフィック・コレクタの蓄積方式は、フロー情報をバイナリデータとして、フラットファイルとして格納する例が多い[11,12]. 本項目では、高速な蓄積機能が要求されるフロー・コンセントレータと多様な検索機能が必要とされるトラフィック・コレクタでそれぞれに適した蓄積方式を評価した.

4.2.1. 評価方法

RDBMS（MySQL,PostgreSQL）,バイナリ形式のフラットファイル,オンラインメモリ DB を用いて、蓄積・検索・削除にかかる時間を測定した. バイナリ形式のフラットファイルについては、開発容易性や新たな情報要素の追加にも対応可能な Perl storable を利用することとした. オンラインメモリ DB については、MySQL の StorageEngine として HEAP を利用することとした.

格納するフローレコードは、NetFlow ver.9 にて配信される情報要素の中から 26 フィールド分を選択し、RDBMS, Perl storable ともに同一の情報要素を含む構造とした. Perl storable の構造は、リファレンスを配列化し、各情報要素については、フィールド名をキー情報とするハッシュ・テーブルをもつ方式とした. この構造のファイルを一定時間単位にセーブし、日付単位に DIR に配置することで、柔軟性のあるフラットファイル方式とした. 図 7 にこの方式を示す.

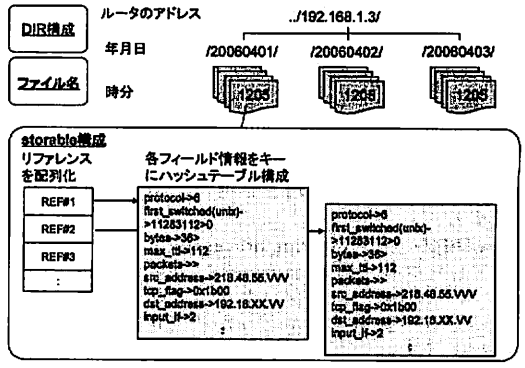

図 7. Perl storable によるフラットファイル方式.

また、蓄積・検索・削除にかかる時間は、各処理の開始・終了時間を測定することで求めている. 検索については、当該全レコードを検索・表示するまでの時間としている. これを図 8 に示す.

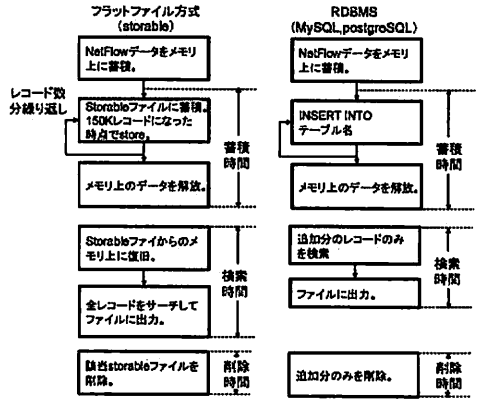

図 8. 測定評価方法.

以下の測定環境にて実施した.

- CPU: Intel(R) Pentium(R) 4 3.40GHz
- メモリ:3574M
- OS: FreeBSD 5.4
- Perl ver. 5.8.6
- MySQL ver. 4.1.16
- PostgreSQL ver.8.0.7

4.2.2. 測定結果

図 9,10 にそれぞれの方式での蓄積・検索に要する時間を示す. 削除処理においては、いずれの方式でも 1 秒前後の処理時間であった. 本結果により、フラットファイル形式が高速に格納可能であり、検索・削除処理においても、他の蓄積方式と同等レベルであることがわかる.

また、フラットファイル方式は、ファイルを細かく配置するため、レコード蓄積数による性能劣化の影響が少ない. これに対して、RDBMS は、蓄積量が増えるに従い検索・削除の性能が劣化するといわれている. 図 11 では、RDBMS（MySQL）の蓄積・検索・削除に

要する時間の蓄積レコード数による影響を示している．大容量化に向けては、約 250K レコードを分単位に挿入し、日単位に 1 日分の約 300M レコードを削除することが要求される．RDBMS は、この点で大容量向けの蓄積方式としては適さないといえる．

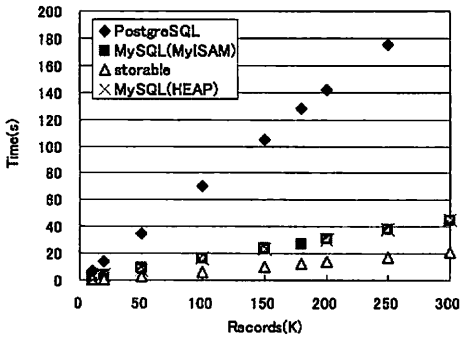

図 9. 蓄積方式によるフロー情報格納時間の比較．

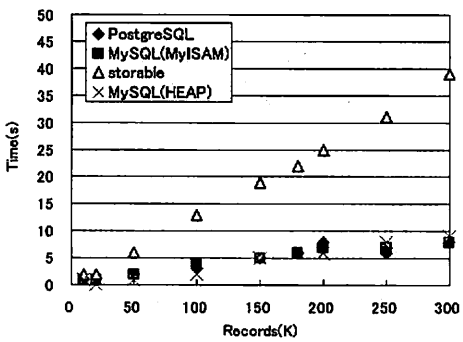

図 10. 蓄積方式によるフロー情報検索時間の比較．

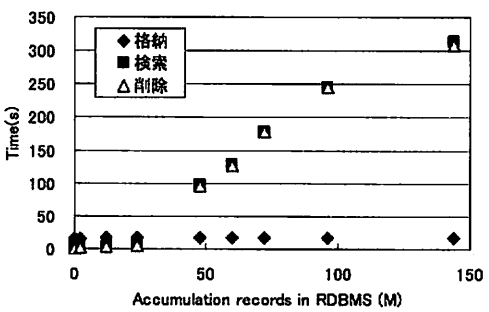

図 11. 蓄積レコード数による性能劣化の影響．
（100K レコードの格納・検索・削除の時間を測定．）

近年、高速性を達成するために提案されているストリーム DB[14]も含めた蓄積方式による比較結果を表 1 にまとめる．

表.1.蓄積方式比較．

蓄積方式	バイナリ形式フラット・ファイル（storable）	RDBMS（MySQL,postgreSQL）	オンメモリDB（MySQL HEAP）	ストリームDB
蓄積速度	◎ 数Mレコード／分	× 数百K レコード／分	◎ 数十M レコード／分	◎ 数十M レコード／秒
検索・削除速度	○ 数Mレコード／分	× 蓄積量に従い遅延	○ 数十M レコード／分	○ 数十M レコード／秒
蓄積容量	○ HDD容量に依存	○ HDD容量に依存	× メモリ容量に依存	○ HDD容量に依存
クエリ機能	× 高度なクエリ機能は実装困難	○ SQL構文を活用可能	○ SQL構文を活用可能	○ SQL構文を活用可能
考察	単純なクエリ機能のみ必要である場合に適する。	蓄積速度が十分でないため不可。	蓄積容量が十分でないため不可。	高度なクエリ機能が必要である場合に適する。

高速にフロー情報を蓄積し、ある時間を指定する程度の単純な検索機能を必要とするフロー・コンセントレータについては、Perl storable のようなバイナリ形式のフラット・ファイルが適している．これに対して、集約・分析など他のアプリケーションなどからの複雑な検索クエリ（問い合わせ）を必要とするコレクタについては、RDBMS もしくはオンメモリ DB などが適している．また、今回評価できなかったが、ストリーム DB については、コンセントレータにて周期毎にランキングを表示するようなアプリケーションを実装する場合には、適していると考えられる．

5. プロトタイプ開発

5.1. プロトタイプ実装

フロー・コンセントレータ構成モデルに基づき、蓄積・選択・集約機能をもつプロトタイプ開発を実施した[14]. 送受信プロトコルは、NetFlow ver.9 を使用し、ヘッダ内の時刻情報についてはルータから配信された情報を維持したまま、集約処理を行うこととしている．

本開発では、集約機能として 5tuple のフロー情報を周期単位に送信元 IP/送信先 IP のペアに集約する機能を実装した．集約機能の柔軟性をもたせるため、集約周期を可変とすることや主要なポート番号については、集約しない機能を付加することとした．これにより、ポート番号 80,25 等の主要なアプリケーションの動向を維持したまま、フロー情報を上位コレクタに配信することを可能とした．

蓄積機能については、フロー情報をバイナリのままダンプする機能と Perl storable による方式の両方式をサポートすることとした．

5.2. プロトタイプによる評価

コンセントレータの有効性を検証するため、コンセントレータの有無により、上位コレクタの負荷の変動を確認している．上位コレクタについては、NW ドメイン全体からフロー情報を収集し、トラフィック量、パケット数などに応じて、トラフィック交流を測定するシステムを配置した[15]. 図 12 に実験構成を示す．3 台のルータに mawi[16]のトラフィックデータを注入し、擬似的に負荷を発生させた．

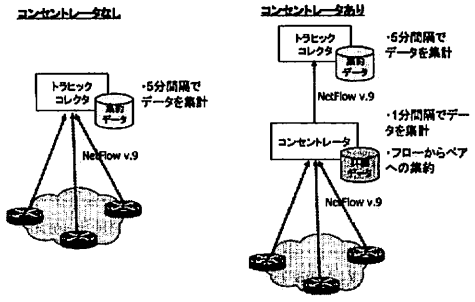

図 12. コンセントレータ評価環境.

図 13,図 14 にコンセントレータあり・なしでの上位トラフィック・コレクタの負荷を示す. コンセントレータでのフロー情報の集約率に依存して、トラフィック・コレクタの負荷が緩和される.

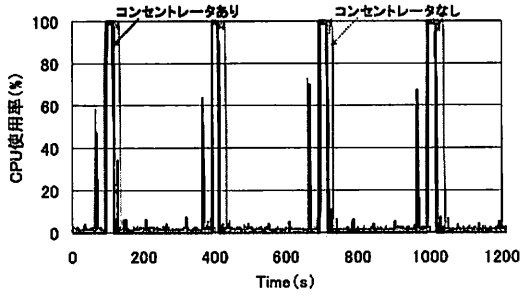

図 13. トラフィック・コレクタの負荷.

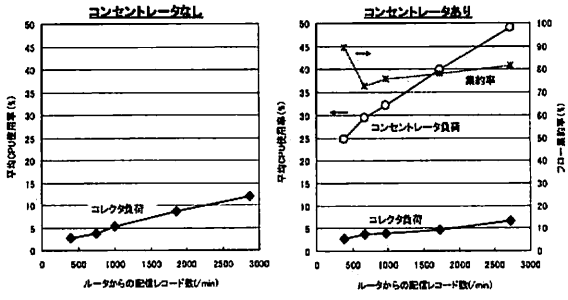

図 14. コレクタ・コンセントレータの平均負荷.

特に、トラフィック交流のようなネットワーク全域から一元的にフロー情報を収集するようなシステムの場合には、コンセントレータによるフロー情報の集約が有益である. 今後は、集約率が高くなる Prefix レンジでの集約などの機能についても実装を検討していく必要がある.

6. まとめ

本稿では、フロー・コンセントレータを用いた大規模トラヒック・コレクタの構成について提案した. また、提案している内部プロセス構造をもとに、フロー・コンセントレータのプロトタイプ開発を行い、その有効性を評価した. 今後は、プロトタイプであるフロー・コンセントレータの実用的な評価と集約・分析機能の多機能化及び更なる高速化を目指していく.

7. 謝辞

本稿は, 総務省委託研究「次世代バックボーンに関する研究開発」による成果である.

文 献

[1] 総務省総合通信基盤局電気通信事業部データ通信課, "我が国のインターネットにおけるトラフィック総量の把握,"2005 年 7 月

[2] B. Claise, "Cisco Systems NetFlow Services Export Version 9", RFC3954

[3] B. Claise, "IPFIX Protocol Specification", draft-ietf-ipfix-protocol-16.txt(work in progress)

[4] B. Claise, "Packet Sampling (PSAMP) Protocol Specifications", draft-ietf-psamp-protocol-02.txt(work in progress)

[5] J. Quittek, T.Zseby, B. Claise, and S. Zander, "Requirements for IP Flow Information Export(IPFIX)", RFC3917

[6] F. Dressler, C. Sommer,G. Munz, "IPFIX Aggregation", draft-dressler-ipfix-aggregation-01.txt(work in progress)

[7] A. Kobayashi, K. Ishibashi, K. Yamamoto and D. Matsubara, "The reference model of IPFIX concentrators", draft-kobayashi-ipfix-concentrator-model-01.txt (work in progress)

[8] R. Enns, "NETCONF Configuration Protocol",draft-ietf-netconf-prot-12.txt(work in progress)

[9] T. Dietz and B. Claise "Definitions of Managed Objects for Packet Sampling", draft-ietf-psamp-mib-05.txt(work in progress)

[10] A. Kobayashi, K. Ishibashi, K. Yamamoto and D. Matsubara, "Managed Objects of IPFIX concentrator", draft-kobayashi-ipfix-concentrator-mib-01.txt (work in progress)

[11] http://silktools.sourceforge.net/

[12] http://nfdump.sourceforge.net/

[13] D. Abadi et al "Aurora: A Data Stream Management System", VLDB Journal Vol.12, No.2, pp120-139,2003

[14] 松原大典, 小林淳史 他, " 次世代バックボーン向けトラヒック情報集約装置の開発" , 2006 年電子情報通信学会総合大会 BS-5-9. 2006 年.

[15] 廣川 裕, 山本公洋 他, " 次世代バックボーン向けトラヒック監視システムの開発" 2006 年電子情報通信学会総合大会 BS-5-11. 2006 年.

[16] http://tracer.csl.sony.co.jp/mawi/