Vol.2021-IOT-54 No.9
2021/7/9

Linux TC(Traffic Control) を用いた監視トラフィックの集約と複数の監視・解析基盤へのトラフィック転送

岩本 裕真[1]　阿部　博[2]　遠峰　隆史[3]

概要：ネットワーク運用者やシステム管理者は、安定したシステム運用の実現やトラブル発生時の原因特定をするために、サーバやネットワーク、セキュリティ機器から出力される監視データの蓄積や解析を行う。代表的な監視データとして、ログやフロー、SNMP があり、それらは Syslog、NetFlow、IPFIX、sFlow、SNMP といったプロトコルで定義されており標準化されている。しかし、多くのネットワーク・セキュリティ機器にはサーバ機器のような高性能な CPU が採用されておらず、システム負荷を上昇させないために監視データの宛先送信数に上限がある。そのため監視データを集約する専用のネットワーク機器やシステムを用いることで、解析基盤に必要な監視データを複製して転送する手法が用いられる。本研究では、ネットワーク機器群から送信された監視データを集約し複数の監視・解析基盤に複製して転送する際に、Linux カーネルのトラフィック制御機能である TC を用いた実証実験を行った。

Traffic replication system with Linux TC (Traffic Control)

Abstract: Accumulating and analyzing monitoring data output from servers, network devices and security equipments is important for network operators and system administrators to ensure stable system operation and identify the causes of problems. Operators are generally used logs, flows, and SNMP as a monitoring data and these mechanisms are defined and standardized in protocols such as Syslog, NetFlow, IPFIX, sFlow and SNMP. Many network and security devices are not equipped with high-performance CPUs like server devices, and thus usually have a limitation to set multiple destination to avoid increasing system loads. Therefore, we use a method to duplicate and transfer the monitoring data required for the analysis infrastructure by a dedicated aggregation system. In this study, we propose to use TC, a traffic function of the Linux kernel, to aggregate the monitoring data from network and security equipments and replicate and transfer the monitoring data to multiple monitoring platforms and data analysis platforms.

1. 背景

ネットワーク運用者やシステム管理者は、システムトラブルの原因を特定するためにサーバやネットワーク、セキュリティ機器から出力される監視データを蓄積し、トラブルの原因となる情報の解析を行う。これらネットワークやシステムを構成する機器の状態を把握し監視するために、監視システムを利用する。監視システムは監視に特化したプロトコルを利用し監視を行う。それらプロトコルの通信方式は Pull 型と Push 型の大きく二つに分類することができる。Pull 型は、監視システムのポーリングルールに定められた時間間隔で、監視システムから定期的にプロトコルで定義される監視パケットで監視対象への問い合わせを行う。Pull 型の実装例として、Ping, SNMP Polling が挙げられる。

Push 型の監視では、監視対象の機器から定期的に監視対象のメトリックデータやリソース状態を監視サーバに送信する。または一定のしきい値を超えた場合にトラップとして、アラートが監視サーバへと送信される。代表的な技術としては Syslog[1], SNMP Trap[2], NetFlow[3], IPFIX[4], sFlow[5] といった実装があげられる。

Pull 型の監視システムは、統合監視システムに用いられることが多く、SNMP Polling を利用し多数の対象を監視することができる。しかしながら Push 型監視は、ネット

[1] 株式会社ブロードバンドタワー
BroadBand Tower, Inc.
[2] トヨタ自動車株式会社
TOYOTA MOTOR CORPORATION
[3] 情報通信研究機構
National Institute of Information and Communications Technology

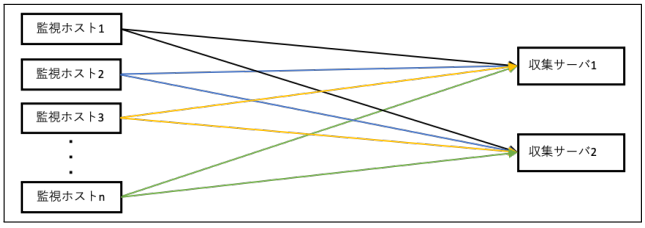

図 1 限定的な送信先設定
Fig. 1 Limited destination settings

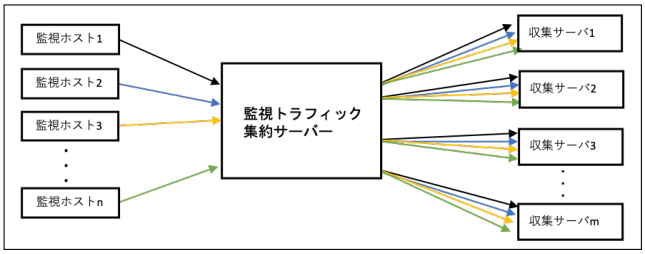

図 2 監視トラフィックの集約イメージ
Fig. 2 Traffic aggregation system

ワーク機器やセキュリティ機器が自発的に監視サーバに対してデータを送信するため、ネットワーク機器やセキュリティ機器のリソース消費の観点から、Push するデータ送信先に上限を設けている。Syslog、SNMP Trap、Flow の送信先設定がこれにあたり、システムに依存するが、図 1 に示すように、1 もしくは 2 つの送信先設定が一般的である。

そのため、監視サーバが複数存在する場合や、監視サーバのアドレス変更が発生した場合には、都度 Push データ送信元であるネットワーク機器やセキュリティ機器の宛先変更が必要となる。また、Syslog や Flow データなどは複数のセキュリティシステムで利用する場合があり、Push データ送信先に上限数があることでデータの分配方法に問題が発生する場合がある。そういった問題を解決するために、Push 型で発生する監視トラフィックを複数箇所に転送する要求がある際には、図 2 のような監視トラフィックを転送するためのシステムにデータを集約し、必要箇所に分配するといった手法が用いられる。

これまで、監視トラフィックの集約機構を持つネットワークアーキテクチャを採用する方法はいくつか提案され実環境で用いられてきた。本研究では、Push 型の監視トラフィックの集約及び転送を汎用的なサーバで実現するため、Linux TC を用いた監視トラフィックの転送基盤の検討及び実証実験を行った。

2. 関連研究

監視トラフィック集約・転送の機能を満たすためにこれまでいくつかの方法が用いられてきた。

システムやネットワーク監視を行う監視システムを運用する際には、各監視プロトコルごとに監視サーバを構築し、監視データの蓄積・解析・可視化を行う。統合監視ではSNMP Trap を扱い、Syslog の蓄積・可視化を行うシステムでは Syslog 収集用に専用システムを構築する。それらプロトコルごとに監視データを扱う専用システムは、収集したデータを別のシステムへと転送する機能が実装されている場合がある。また、監視プロトコルごとの専用システムで行うデータの転送方法以外に、パケットを複製するネットワークアプライアンスやソフトウェアが存在する。本節では既存手法の事例を挙げる。

2.1 SNMP Trap 転送

snmptrapd は、Net-SNMP が提供及び開発を進めており、多くの UNIX 系 OS の Simple Network Management Protocol(SNMP) 実装として利用されている。snmptrapd の実装では、SNMP Trap メッセージを別のサーバに転送することが可能である [6]。

2.2 Syslog 転送

Syslog 受信デーモンである Rsyslogd は、外部の Syslog サーバから受信した Syslog を他のサーバへと転送することが設定で可能である。[7]

```
*.* @192.168.0.1
or
*.* @192.168.0.1:任意のポート
```

この設定により、転送先ホストである「192.168.0.1」の UDP 514 ポートへデフォルト設定で転送を行うことができる。また任意の UDP 転送先ポートへの転送設定も可能となる。「@」を「@@」にすることで TCP での転送も可能となる。同様に商用製品である syslog-ng にも、Syslog を他のサーバへと転送する機能が実装されているが、こちらは転送条件に Rsyslogd よりも細かいルールを設定することが可能である。

2.3 Flow データの転送

ntop 社が提供する nProbe[8] では、Flow に対応していないルータに接続し、通信トラフィックから NetFlow/IPFIX データを生成することが可能な製品である。nProbe には Proxy mode と呼ばれる機能があり Flow を収集するシステムである Flow コレクタに対しフロー転送を Proxy する機能が存在する。

2.4 Network Packet Broker(NPB)

NPB は、ネットワーク機器やセキュリティ機器からミラートラフィックを生成可能なポートミラーリングを用いるか、またはネットワーク Test Access Point(TAP) デバイスで収集したトラフィックデータを、監視トラフィックが必要なセキュリティシステムやモニタリングシステムに複製し転送するための専用機器である。

パケットのカプセリング処理やヘッダ付加や削除、パケット重複排除の機能が実装されている機器もあり、専用のハードウェアチップを搭載している事もあるため高い性能を発揮する。NPB 製品としては、Gigamon 社 [9] の Gigamon や Keysgiht 社の VisionOne[10], VisionX が広く知られている。NPB に関する先行研究 [11] では、Arista 社の DANZ Monitoring Fabric (DMF) をキャンパス・ネットワーク・コンポーネントで利用した場合に約 15,000 ドルの導入コストかかる事が報告されている [12]。

2.5 Lagopus

Lagopus は、オープンソースソフトウェアとして公開される、OpenFlow1.3 をベースとしたソフトウェアスイッチ・ルータである。5-Tuple を元にデータ転送を可能な OpenFlow の仕様加えてパケットのコピー機構を実現している。データプレーンの転送処理は Data Plane Development Kit(DPDK)[13] で実装されており汎用なサーバにおいても高速にパケット転送処理が可能である。[14]

3. 既存手法の課題

ネットワーク運用時に管理者がトラブル対応や監視データを確認したい場合、トラブルの元となる対象ホストの検索を行う場合がある。対象ホストがドメインネームシステムと連動しない送信元 IP アドレスで示される場合、IP アドレスを用いることによりホストを一意に識別・検索することが可能となる。

各監視プロトコルごとに監視システムを構築しそのシステムで転送を行う場合には、送信元アドレスは、最初に監視データを送信したホストの IP アドレスから、転送を行う各監視プロトコルのシステムの送信元アドレスに書き換えられてしまう。監視トラフィックを送信した送信元ホストを一意に識別するために用いたかった送信元 IP アドレスが、書き換えられたことにより一意性を保てなくなってしまう。

この問題を回避するために、転送する際に各監視プロトコルのメッセージに送信元 IP アドレス情報や一意に識別可能な ID を割り当てる事も可能ではあるが、各監視プロトコルのシステム単位に個別設定を行う必要があり、また転送先システムでも識別情報を処理するために工夫が必要になる。

また複数のネットワークセグメントから監視トラフィックを受信したいという要求も存在する。その場合には、各収集サーバがすべてのネットーワークに対して、物理的、または論理的にネットワークの接続を確保する必要がある。

NPB はパケット処理専用機のため、専用プロセッサを備え高い性能を発揮するが、汎用的サーバで同等の処理を実現する場合と比較して導入コストが大きくなる。

DPDK を利用する Lagopus は、汎用サーバやスイッチを利用する観点から NPB を利用した場合と比較してコスト面の利点がある。しかしながら、ソフトウェアの仕組みである DPDK フレームワークを利用するため、DPDK 自体に不具合やセキュリティ問題が見つかった場合にはソフトウェアのアップデートを行う必要がある。ソフトウェアスイッチに関しても同様の問題を抱えており、ソフトウェア保守の観点からもソフトウェアのアップデートに関する継続性に関して監視運用に大きく影響が出る可能性がある。

4. 提案手法

3 節で述べた課題を解決するために、本研究では Linux iproute2 で提供されている Traffic Control(TC) を用いて監視トラフィックの集約・転送機構である Logfwdr を実現をした。本節では、Linux TC を用いるに至った理由を述べる。

4.1 Linux TC を用いたの監視トラフィック転送

TC は Linux カーネルのネットワーク制御と監視を行うためのユーザ空間ユーティリティの一つである。TC では、カーネル上に qdisc(Queueing Discipline) と呼ばれるパケットを格納するためのオブジェクトを定義する。送受信したパケットを qdisc を用いることでパケットの書き換えをカーネルに組み込む事が可能である。受信した監視トラフィックをプロトコルごとに分類するため図 3 で示した IngressNIC でパケット処理を行った。qdisc はパケットの転送以外に、複雑な設定を行うことができるが、本研究では複雑なトラフィック制御を行わず MAC アドレスや IP アドレスの書き換えと複製、転送のみを行うためシンプルなクラスレス qdisc に属する ingress を用いた。[15]

Ingress NIC で受信したパケットを、監視トラフィックの宛先ポート番号でフィルタリングし、Egress NIC の MAC アドレスを送信元 MAC アドレスに変更する。

4.2 転送機構としての Linux TC の選定理由

単純な Linux Socket をベースとしたユーザーランドプログラムでパケット処理を行うアプリケーション実装では、パケットの受信時にカーネルメモリ空間上のメモリ空間からユーザーランドのメモリ空間へメモリコピーが必要となるためオーバーヘッドが生じるため、高速な処理能力が期待できない。送信時はユーザーランドのメモリ空間からカーネルメモリ空間へコピーが生じる事で同様にオーバヘッドが生じる。

そのため高速にパケット処理を行うためには DPDK や netmap[16] といった技術が用いられることがある。DPDK は、カーネル機能をバイパスする事によりシステムコール経由せず直接ハードウェアを制御することができるためユーザーランドプログラムと比較すると高速化が可能となる。netmap は、パケット送受信用のバッファを確保し

mmap を用いてアドレス空間をマッピングすることでユーザーランド空間とカーネル空間のメモリコピーを減らし高速化している。しかしながら、DPDK や netmap といった技術は、低レイヤのネットワークプログラミング技術が必要でありプログラムの記述難易度が高い。TC は DPDK や netmap と比較すると、プログラミング言語ではなくシェルコマンドで簡単に実行できる。

また iproute2 は Linux の標準的なネットワーク制御を行うためのユーティリティ群として提供されているため、ソフトウェアの保守性という観点でも選定の理由として大きい。本研究では、ソフトウェアの保守性や柔軟なパケットマッチングと処理機構を実現可能なことから、比較的簡易に転送機構を構築する事が可能であるので TC を選択した。

5. 設計・構築

4.1 に述べた理由より、本研究では、2021 年 4 月に開催された Interop Tokyo 2021 ShowNet において、TC を活用したシステムである Logfwdr を実現した。ShowNet[17] とは、ネットワーク・セキュリティ・サーバ機器の相互接続を行うイベントである。本節では ShowNet において Logfwdr 活用した際の物理・論理設計について述べる。

5.1 ShowNet とは

ShowNet は、年に一度、幕張メッセ展示場と国際会議場に構築、運用されるデモンストレーションネットワークの名称である。ShowNet は、産学関係者が集まり最新鋭の製品・サービスを用いて Interop の由来であるインターオペラビリティ（InterOperability：相互接続性）を確認しながら、次世代の ISP バックボーンやキャリアネットワーク、データセンターネットワーク、エンタープライズネットワークを模した環境を構築する。ShowNet では、HotStage と呼ばれる事前検証期間 1 週間と、Deploy と呼ばれる展示会の準備期間 4 日間、展示会 3 日間の合計 2 週間で構築が進められる。ShowNet は、単にデモンストレーションネットワークとして構築されるのではなく、Interop に出展している企業ブースのインターネット接続や来場者向け Wi-Fi ネットワークとして提供される。

5.2 ShowNet における監視環境

ShowNet には、バックボーンネットワークを構築するためのルータ機器類、データセンターネットワークに用いられるルータ・スイッチ群及びそれらが接続されるサーバ群など多種多様な機器が接続される。さらに ShowNet は、来場者及び ShowNet 関係者に提供される Wi-Fi 関連機器や温湿度センサーといった機器、クラウド環境の仮想マシン群やクラウドサービスなどによって構成される。これら全てのネットワークや機器、サービスが監視対象となる。2021 年の実績では、監視対象の総ホスト数は 455 台となった。

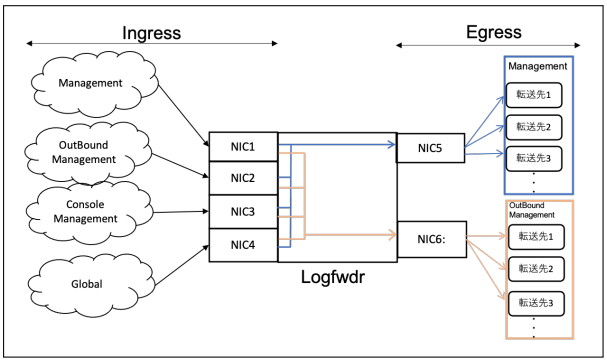

図 3 Logfwdr のネットワーク接続
Fig. 3 Network connection for Logfwdr

表 1 Logfwdr のマシンスペック
Table 1 Machine Spec

機器名	SuperServer E300-9D-8CN8TP
CPU	Intel Xeon processor D-2146NT 8 Cores, 16 Thread
Momory	64GByte
OS	Ubuntu 20.04.2 LTS
version	5.4.0-70-generic

本研究で対象となる ShowNet 環境で利用された Push 型の監視トラフィックとして、Syslog, SNMP Trap, sFlow/IPFIX/NetFlow が用いられ、これら対象となるトランスポート層のプロトコルは全て UDP を利用した。一部例外として、TCP で送信される Push 型の監視トラフィックが存在するが Logfwdr で TCP を終端し、UDP にプロコトルを変換してから送信を行った。

5.3 ネットワーク構成

Logfwdr が監視トラフィックを受信するためのネットワークセグメントは、ShowNet 関係者が機器の管理用セグメントとして利用する Management ネットワーク、一部 Firewall 越しにインターネットと接続可能な Outbound-Management ネットワーク、コンソール接続用に分離された ConsoleManagement ネットワーク、そして一部のバックボーンルータからの IPFIX トラフィックを受信するための Global ネットワークの計 4 つ定義されており、全てのセグメントから監視トラフィックを受信した。そのため Logfwdr には、監視トラフィックを受信するための Ingress 側の NIC を 4 つ用意し、監視トラフィックの転送先用 NIC として Egress 側に転送先のサブネット数 2 つ用意した。

図 3 に構成図を示す。Logfwdr で本構成を実現するために、複数の NIC を組み込むサーバが必要となったため Xeon-D の SoC を搭載した表 1 にスペックを示す、SuperMicro 社の SuperServer E300-9D-8CN8TP を採用した。Logfwdr が受信した監視トラフィックの転送先機器・システムを表

表 2　Logfwdr の転送先リスト
Table 2　Logfwdr's Forwarding List

転送先システム名	機能	Syslog	SNMP Trap	sFlow	NetFlow	IPFIX
Zabbix	統合監視	✓	✓			
SystemAnswer Logoption	統合監視	✓	✓			
Flowmon collector	Flow コレクタ			✓	✓	✓
SystemAnswer Statsoption	統合監視	✓				
Hayabusa	ログ解析	✓				
logdispatcher	セキュリティ	✓				
nirvana-flow	セキュリティ			✓	✓	✓
nsx-ndr	セキュリティ			✓	✓	✓
Skybox	セキュリティ	✓				
Stealthwatch	セキュリティ			✓	✓	✓
Thunder 7445-TPS	DDoS 対策			✓		
Paragon Insight	ネットワーク分析	✓		✓		
VMware LogInsight	ログ解析	✓				

2 に示す。転送先機システムがどのどのような機能の製品であるかを示し、転送を必要とするプロトコルの対応をす。例として、統合監視システムの Zabbix は Syslog と SNMP Trap を転送データとして必要とする。

5.4　Logfwdr の設定

監視トラフィックで用いられるトランスポート層プロコルは UDP であるため、TCP のように End-To-End でセッションを維持することはできない。Logfwdr が各監視システムにパケットを転送する際の動作は図 4 のように表すことができる。

(1) 監視対象となるホストから送信されてきたパケットの宛先ポート番号でフィルタリングを実施
(2) `skbmod` を用いて宛先 MAC アドレスを転送先の MAC アドレスに書き換え
(3) `action nat` を用いて宛先 IP アドレスの書き換え
(4) `action mirror` を用いて転送先ネットワークに属する NIC を指定し送信

以上の動作を転送先のホスト数と対象トラフィックのプロコル数に応じてチェーンルールとして繋げる。

動作の流れを図 5 に示す。Ingress NIC に入ったパケットは、パケットフィルタ通過後に `action flow` を N 個接続し、Egress NIC からパケットを出力する。

具体的に TC の設定を行うコマンドは以下となる。

```
# tc filter add dev eno1 parent ffff:
    protocol ip prio 10 u32 match ip dport
    転送対象のポート番号 0xffff \
action skbmod set smac 転送元MAC アドレス set
    dmac 転送先MAC アドレス pipe \
action nat ingress  転送元 IP アドレス 転送先
    IP アドレス pipe \
action mirred egress mirror dev eno2
```

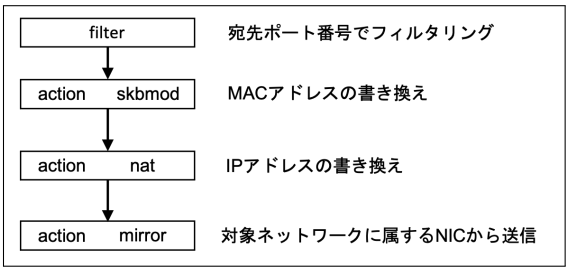

図 4　TC のアクションフロー
Fig. 4　action flow

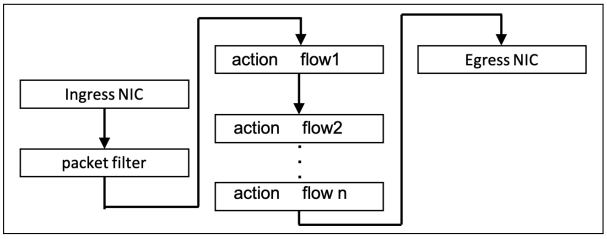

図 5　action の連鎖
Fig. 5　chain of actions

このコマンド例では、eno1 のインタフェースから入力された ingress のデータを、`action skbmod` で MAC アドレスの書き換えを行い、`action nat ingress` で IP アドレスの書き換えを実施、最後の `action mirred egress` でミラーパケットを eno2 のインタフェースから出力する。

6.　評価

5.1 節で述べたように ShowNet 期間 2 週間の検証の結果から Logfwdr の CPU とメモリのリソース消費の観点で評価を行う。まず記録している全期間 4/7 から 4/16 の CPU LoadAvarage のグラフ図 6 を表す。1~2 時間に一度不定期的にスパイク状に LoadAvarage 数が上昇している事が観測された。スパイク形状に上昇している部分でも LoadAvarage 数は 3.0 に満たない値であった。そのため今回用意したマシンの CPU では物理で 8 コアのキャパシティがあり CPU リソースに関しては十分に余裕のある環境であったと推察することができた。

次に全期間 4/7 から 4/16 の Memory Total Free のグラフ図 6 を表す。メモリに関しては、全期間を見てもほとんど変化は見られる事がなくトラフィック転送がメモリ消費に影響があるようには観測できなかった。4/14 17:00 頃から 4/15 11:00 頃までメモリ使用量が上昇したことにより使用可能な全体メモリ数が段階的に減少している事が観測されたが、その後定常状態近くの値まで戻った事が確認された。

転送性能に関しては、Egress の NIC カウンタ値が取得できなかったため今回定量的な比較ができなかった。

Syslog の受信に関しては、VMware LogInsight で受信した Syslog 数は、最大で 217,507,130 messages/日であり全期間で 851GByte の容量が蓄積された。1 日単位の集計値

表 3　全日程で集計した Syslog 数
Table 3　Syslog message counts

日程	Syslog Message 数	日程	Syslog Message 数
4/2	16,131	4/10	172,526,140
4/3	493,475	4/11	230,187,569
4/4	1,153,175	4/12	149,412,196
4/5	15,287,717	4/13	160,628,712
4/6	8,902,871	4/14	196,523,872
4/7	34,416,993	4/15	183,863,935
4/8	112,360,864	4/16	141,905,840
4/9	217,507,130		

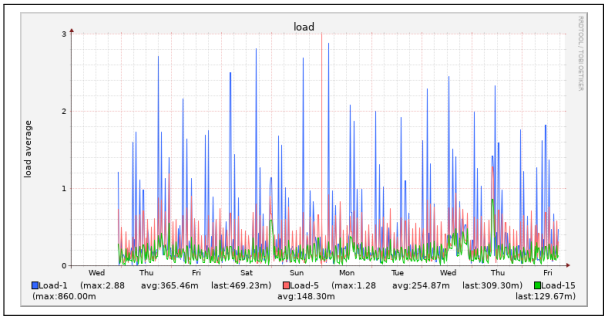

図 6　ShowNet2021 全期間 CPU LoadAvarage の推移
Fig. 6　CPU LoadAvarage for the entire period of ShowNet2021

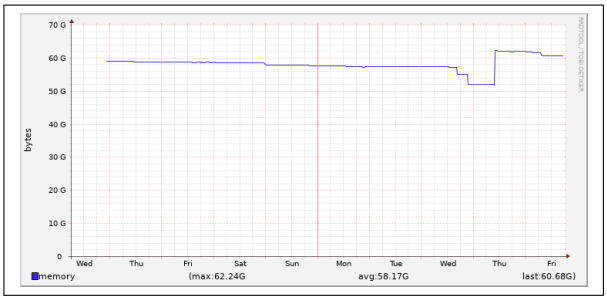

図 7　ShowNet2021 全期間 Memory Total Free の推移
Fig. 7　Memory Total Free for the entire period of ShowNet2021

を表 3 に示す。Hotstage 初期より Syslog の転送が行われ、日程の後半には平均数千行の Syslog が毎秒転送された。日により Syslog 数が増減するのは、転送先機器モニタリング・セキュリティ機器が転送される Syslog の量に対応できず、ログ出力元のセキュリティ装置のログレベルを調整したからである。

また SNMP Trap の受信に関しても大量のパケットを転送しており、1 日に最大 7,466,628 messages のトラップが観測された。

7.　考察

6 節の結果から、Logfwdr は CPU とメモリ利用率が小さく、汎用サーバ上で動作させるには十分な性能が発揮できると考えられる。性能面でも、Syslog の転送に関しては、ログを失うことなく日に最大 2 億行を超えるログを転送し続け、SNMP Trap に関しても、日に 700 万を超える数を転送した。さらに Flow も同時に転送していることから、Logfwdr はメモリリークもなく低いロードで複数の監視プロトコルを十分な転送性能で実現できたと言える。

DPDK を用いて転送を実現する場合には、特定の CPU コアを転送アプリケーションに割り当て占有化を行う。これにより、DPDK アプリケーションに割り当てらた CPU コアの負荷は上昇し、CPU コアの占有化によりシステム全体として利用可能な CPU コアの数が減少する。転送以外のアプリケーションをシステムで動作させる場合には、コアの占有は発生しないため Linux TC での転送方式に利点が発生する可能性がある。

運用的な観点では、Linux TC に転送ルールを設定する場合に、送信先 MAC アドレスと送信先 IP アドレスを tc コマンドを用いてルールチェーンとして設定していく。これは、コマンドを投入するだけで設定を行うことができる運用上の利点となりうる。しかしながら転送先に追加や削除、転送プロトコルの変更や追加などの転送ルールの変更が発生した場合、現在投入している tc のルールを一度破棄して、再度コマンドを投入しルール設定を行う必要があるため、設定変更を行うたびに転送の瞬断が発生する。ルール変更が行われない環境であれば影響は少ないが、高頻度に設定変更が行われる環境ではなにかしらの対応が必要となる。

コスト面で考えた場合、転送、複製の専用機である NPB の導入にかかるコストと比較し、小型汎用サーバで転送システムを実現した場合には、1/10 から 1/100 程度の予算で機能を実現できる可能性がある。今回利用した実験機は、比較的安価な Xeon-D 搭載の SoC マシンであり、NIC の数が最初から多いものを選択したため、実際に NPB とのコスト差を考えると 1/50 程度である。もちろん NPB の全ての機能を実装したわけではなく、一概に比較することはできないが組織の規模や転送するデータ量によっては、小さな組織であれば十分に現実的な選択肢となりうる。

8.　まとめと今後の課題

本研究の実証として、ShowNet 会場での 2 週間の動作実験を行った。今回は ShowNet の性質上、2 週間という実験期限の制限があったが、Logfwdr が実際の運用システムに組み込まれ利用される場合にはさらなる中長期での動作テストが必要と考える。今回、TC の仕様により同一のポート番号において転送先ホストの数が 7 台の上限となった。上限が存在する理由について、TC の仕様上どのような理由により上限数が決まっているのか OS レベルで深く追求していく必要がある [18]。また ShowNet での運用では、2 台の Logfwdr の冗長化自動切り替えを実施せず、Primary/Backup 構成を取り、Logfwdr 障害時には手動で

切り替えを実施する必要があった。障害時の冗長化自動切り替えに関しては、ルータやスイッチで用いられる Virtual Router Redundancy Protocol(VRRP) を利用することで実現可能であると考え今後の課題とする。

また、TC にはハードウェアオフローディングによる処理が可能な Smart NIC も存在し、よりハイパフォーマンスな転送機構を実現できる可能性がある [11]。プログラマブルデータプレーンの応用という観点では、Field Programmable Gate Array(FPGA) による専用の転送回路の設計や、P4(Programing Protocol-independent Packet Processing) 言語 [19][20] を用いた HDL(Hardware Description Language) を用いない FPGA の利用など SmartNIC の積極的採用により、Logfwdr と同様の機能が実現可能である。

負荷試験の継続と、冗長化の実現、更なるオフローディングの追求を行い、今後の ShowNet での Logfwdr の安定運用を実現したい。

参考文献

[1] Gerhards, R.: The Syslog Protocol, RFC 5424 (2009).
[2] Wijnen, B., Harrington, D. and Presuhn, R.: An Architecture for Describing SNMP Management Frameworks, RFC 2571 (1999).
[3] Claise, B.: Cisco Systems NetFlow Services Export Version 9, RFC 3954 (2004).
[4] Aitken, P., Claise, B. and Trammell, B.: Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information, RFC 7011 (2013).
[5] Panchen, S., McKee, N. and Phaal, P.: InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks, RFC 3176 (2001).
[6] : Manpage of SNMPTRAPD, http://www.net-snmp.org/docs/man/snmptrapd.html.
[7] : rsyslog.conf(5) — Linux manual page, https://man7.org/linux/man-pages/man5/rsyslog.conf.5.html.
[8] : nProbe™ An Extensible NetFlow v5/v9/IPFIX Probe for IPv4/v6, https://www.ntop.org/products/netflow/nprobe/.
[9] : Gigamon, https://www.gigamon.com/.
[10] : Keysight Vision ONE, https://www.keysight.com/us/en/products/network-visibility/network-packet-brokers/visionone.html.
[11] Kim, H., Chen, X., Brassil, J. and Rexford, J.: Experience-driven research on programmable networks, *ACM SIGCOMM Computer Communication Review*, Vol. 51, No. 1, pp. 10–17 (2021).
[12] : Arista DANZ monitoring fabric., https://www.arista.com/en/products/danz-monitoring-fabric.
[13] : Home - DPDK, https://www.dpdk.org/.
[14] : Lagopus switch and router, https://www.lagopus.org/.
[15] Salim, J. H.: Linux Traffic Control Classifier-Action Subsystem Architecture, *Proceedings of netdev 0.1* (2015).
[16] Rizzo, L.: netmap: a novel framework for fast packet I/O, *21st USENIX Security Symposium (USENIX Security 12)*, pp. 101–112 (2012).
[17] InteropTokyo: ShowNet とは — Interop Tokyo 2021 (インターロップ東京 2021), https://www.interop.jp/shownet/.
[18] Horman, S.: TC Flower Offload, *NetDev Conference* (2017).
[19] : Specifications P4; Language Consortium, https://p4.org/specs/.
[20] Bosshart, P., Daly, D., Gibb, G., Izzard, M., McKeown, N., Rexford, J., Schlesinger, C., Talayco, D., Vahdat, A., Varghese, G. et al.: P4: Programming protocol-independent packet processors, *ACM SIGCOMM Computer Communication Review*, Vol. 44, No. 3, pp. 87–95 (2014).