情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2011-OS-119 No.3
Vol.2011-EMB-23 No.3
2011/11/29

並列NIDS構築に向けた ソフトウェアロードバランサのパケット転送性能改善

山田正弘[†1] 阿部洋丈[†1] 市川昊平[†1]
伊達進[†1] 下條真司[†1,†2]

近年，ネットワークを利用するサービスに対する攻撃が増加および多様化しており，NIDS (Network-based Intrusion Detection System) の重要性が高まっている．今日の NIDS にはネットワークの広帯域化に応じた高速な処理が求められており，高性能な NIDS の実現が急務である．このような背景から，本論文ではロードバランサを用いて NIDS の処理を並列化する並列 NIDS の実現を目的とし，高性能なソフトウェアロードバランサを設計，実装する．提案実装では高速なパケットキャプチャおよび転送モジュールである PF RING を利用し，パケット転送処理をマルチスレッドで並列化することで高速なパケット転送を実現する．さらに，本論文の性能評価によって，提案実装におけるパケット転送が 9.8Gbps を超えるスループットを達成し，既存研究のソフトウェアロードバランサで採用されているパケット転送よりも高い性能を持つことを示す．

Performance Improvement of Packet Forwarding in Software-based Load Balancer for Parallel Distributed NIDS

Masahiro Yamada,[†1] Hirotake Abe,[†1]
Kohei Ichikawa,[†1] Susumu Date[†1]
and Shinji Shimojo [†1,†2]

With the increase and diversity of attacks against network services, NIDS (Network-based Intrusion Detection System) has been taking a role of importance. For the traffic analysis of the today's high-speed network, the demand on high performance NIDS has been increased. From the background, we focus on parallel distributed NIDS that analyzes network traffic on multiple PCs in a high-throughput fashion. In this paper, we design and implement a high performance software-based load balancer that distributes the workload of traffic analysis to multiple commodity PCs. A prototypic implementation of the proposed software-based load balancer has realized high-throughput packet forwarding by multi-threaded processing and PF RING which provides high-speed packet capturing and sending. Our experiments indicate that, packet forwarding in the proposed software-based load balancer has achieved 9.8 Gbps throughput.

1. はじめに

近年，ネットワークを利用する様々なサービスの普及に伴い，これらのサービスに対する攻撃が増加および多様化している．同時に，これらの攻撃からネットワークやネットワークに接続するコンピュータを保護するためのネットワークセキュリティ技術の重要性が高まっている．このような技術として主要なものに NIDS (Network-based Intrusion Detection System)[1] が挙げられる．NIDS は組織内部ネットワークとインターネット間を流れるトラフィックを取得して解析することで，異常なトラフィックや攻撃を検知して管理者に通知するシステムである．

一方，今日のネットワーク技術の発達から 10Gbps の広帯域ネットワークが普及してきており，ルータやスイッチ，ファイアウォールなどのネットワークシステムの高速化[2] が求められている．NIDS もネットワークの広帯域化に伴い高速化が求められているシステムであり[3),4]，大量のトラフィックデータをリアルタイムに解析する必要がある．しかし，Snort[5] や Bro[6] などの主要な NIDS ソフトウェアは汎用 PC 向けに開発されており，1 台の汎用 PC の計算能力では 10Gbps のトラフィックを解析することが困難である．そこで，今日では，多くの広帯域ネットワーク向けの NIDS は FPGA (Field-Programmable Gate Array) や CAM (Content-Addressable Memory) などの高価なハードウェアを用いて実装されており[7),8]，コストが高い．これに加えて，これらのハードウェアで実装される NIDS は柔軟性が低く，主要な NIDS ソフトウェアでサポートされている TCP コネクションの再構築によるアプリケーションレイヤでの解析[9)–11)] の実現が困難である．このため，ある TCP コネクションにおいて攻撃者が攻撃を意図的に複数のパケットに分割することで，NIDS による検知を回避されてしまう恐れがある．

これらの問題を解決するため，汎用 PC ベースの複数の NIDS (以下，NIDS ノード) で

†1 大阪大学
Osaka University
†2 独立行政法人情報通信研究機構
NICT

Vol.2011-OS-119 No.3
Vol.2011-EMB-23 No.3
2011/11/29

解析を並列化する並列 NIDS[12)–14)] に関する研究がある．並列 NIDS はロードバランサによって監視トラフィックを分割して各 NIDS ノードに割り当てることで解析を並列化する．ロードバランサを汎用ハードウェアベースのソフトウェアロードバランサとして実装することで，比較的低コストで並列 NIDS を構築できる．さらに，ソフトウェアロードバランサにおいて，IP アドレスやポート番号によって定義されるコネクションに基づいてトラフィックを分割することで，各 NIDS ノードで TCP コネクションの再構築が可能になり，アプリケーションレイヤでの解析を実現できる．既存研究では，解析対象とするネットワークの帯域や想定されるトラフィック量に応じて，NIDS ノードの数をスケーラブルに変更できる並列 NIDS のアーキテクチャが提案されている[13),14)]．一方で，このようなアーキテクチャでは，ネットワークの帯域やトラフィック量に関わらず，ソフトウェアロードバランサが全てのトラフィックを処理することが前提とされている．このため，ソフトウェアロードバランサが全てのトラフィックを処理できず，パケットロスによって検知漏れが発生するという問題がある．よって，広帯域ネットワークに向けた並列 NIDS を低コストで実現するためには，ソフトウェアロードバランサの性能に関する検証が必須である．しかし，既存研究では，著者の知る限りで，ソフトウェアロードバランサの性能に関する検証が十分になされていない．

このような背景から，本論文では，10Gbps ネットワークにおいてアプリケーションレイヤでの解析を行う汎用ハードウェアベースの並列 NIDS を実現することを目的とし，ソフトウェアロードバランサを設計，実装する．さらに，提案する実装のパケット転送性能を評価し，既存研究のソフトウェアロードバランサで採用されているパケット転送よりも高いスループットを達成することを示す．以下，2 節では並列 NIDS に向けたソフトウェアロードバランサの要件および設計について述べる．3 節で，提案するパケット転送の実装について述べた後，4 節では，提案するパケット転送と既存のパケット転送の性能を比較評価する．5 節では，今後の課題とまとめについて述べる．

2. 並列 NIDS に向けたソフトウェアロードバランサの要件

並列 NIDS において，ソフトウェアロードバランサは監視トラフィックを分割して各 NIDS ノードに割り当てる役割を持つ．さらに，いくつかの既存研究では必要に応じた負荷分散機能も提案されている[12),15)]．これらの機能を持つソフトウェアロードバランサの例を **図 1** に示す．以降では，10Gbps ネットワークでアプリケーションレイヤでの解析を実現する並列 NIDS のソフトウェアロードバランサ構築に向けて，ソフトウェアロードバランサのトラフィック分割手法，負荷分散，性能に関する 3 つの要件を整理し，その後にそれぞれに関する要件を満たすための設計方針を説明する．

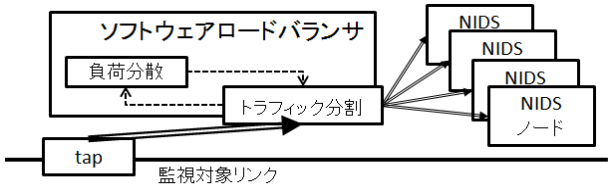

図 1 ソフトウェアロードバランサ
Fig. 1 A software-based load balancer.

2.1 ソフトウェアロードバランサの要件

トラフィック分割

一般的に，並列 NIDS には解析対象とするネットワークの帯域や想定されるトラフィック量に応じて NIDS ノード数をスケーラブルに変更できるという特徴があり，ソフトウェアロードバランサにおいても NIDS ノード数を制約しない構成およびトラフィック分散手法が求められる．これに加えて，低コストで並列 NIDS を実現するために，ソフトウェアロードバランサは最小限の汎用ハードウェアによって構築する必要がある．

負 荷 分 散

並列 NIDS では，一般的に，ソフトウェアロードバランサが IP アドレスやポート番号に基づいてトラフィックを分割し，各 NIDS ノードにおいて TCP コネクションを再構築することでアプリケーションレイヤでの解析を実現する．各 NIDS ノードは汎用 PC で実装されるため，単位時間に解析できるトラフィック量に限りがあり，一定量以上のトラフィックは解析できずに破棄される．NIDS ノードにおいてトラフィックの破棄すなわちパケットロスが発生した場合，検知漏れにより侵入検知の機能を損なう可能性があるため，高いスループットのトラフィックに対してもパケットロスは最小限にしなければならない．このため，ソフトウェアロードバランサにおいて，特定の NIDS ノードに対して解析可能な限度を超えるトラフィック量を割り当てないような負荷分散機能が必要である．

性　　能

NIDS は他のインライン型のネットワークシステムと異なり，コピーされたトラフィックを取得して解析するシステムである．このため，レイテンシやジッタの性能が侵入検知に与える影響は少なく，これらの性能はソフトウェアロードバランサの要件として特別に考慮す

Vol.2011-OS-119 No.3
Vol.2011-EMB-23 No.3
2011/11/29

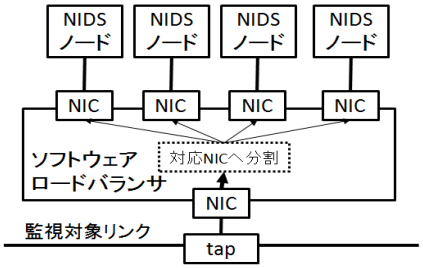

図 2 トラフィック分割手法 1
Fig. 2 Method 1.

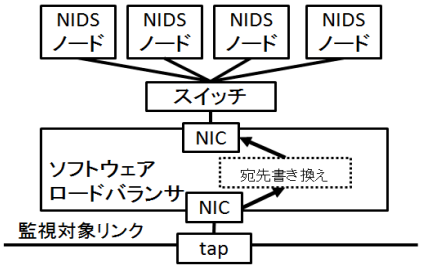

図 3 トラフィック分割手法 2
Fig. 3 Method 2.

る必要はない．

一方で，前節で述べたように，ソフトウェアロードバランサが並列 NIDS のボトルネックとならないために，高いスループットでのトラフィック処理が必要である．さらに，ソフトウェアロードバランサにおけるパケットロスは，NIDS ノードにおけるパケットロスと同様に侵入検知の機能を損なう可能性がある．よって，ロードバランサには最小限のパケットロスで高いスループットのトラフィックを処理できるアーキテクチャが求められる．

2.2 トラフィック分割

ソフトウェアロードバランサにおけるトラフィック分割手法として，**図 2**，**図 3** に示す 2 つの手法が挙げられる． 図 2 の手法はソフトウェアロードバランサに各 NIDS ノードと 1 対 1 で対応する NIC (Network Interface Card) を実装し，割り当て先の NIDS ノードに対応する NIC を介してパケットを転送する方法である．この手法では，NIDS ノード数に応じて複数の汎用 PC でソフトウェアロードバランサを実装する必要がある．複数の汎用 PC でソフトウェアロードバランサを実装する場合，複数の PC 間での状態共有が必要である．図 3 の手法はパケットのヘッダ情報を書き換えることで，割り当て先の NIDS ノードをパケットの宛先として指定する方法である．この手法は，まず，ロードバランサがトラフィックのパケットの宛先 IP アドレスや宛先 MAC アドレスの書き換えなどによってパケットの宛先を書き換えて，汎用の L2 スイッチに転送する．次に，汎用の L2 スイッチが書き換えられた宛先に従って特定の NIDS にパケットを割り当てることでトラフィックを分割する．この手法では，汎用 L2 スイッチに実装されたポート数の範囲であれば，ハードウェアを追加することなく NIDS ノード数をスケーラブルに変更できる．さらに，汎用 L2 スイッチに実装されたポート数を上回る NIDS ノード数に対しても，汎用 L2 スイッチの追加によって対応することができるため，状態共有などの追加機能は必要ない．

前述の要件を考慮して，手法 1 と手法 2 を比較すると，最小限のハードウェアで NIDS ノード数をスケーラブルに変更できることと，複数の PC 間での状態共有が必要無いことから，本研究では手法 2 をトラフィック分割に採用する．

2.3 負荷分散

NIDS ノードの負荷分散を実現するためには，各 NIDS ノードに割り当てているトラフィック量を監視し，NIDS ノードで解析可能なトラフィック量の限界値を超えないようにソフトウェアロードバランサが調整すればよい．各 NIDS ノードに割り当てているトラフィック量を監視する方法は以下の 2 つが挙げられる．監視方法 1 は各 NIDS ノード上に監視モニタを設置することで NIDS ノード毎にトラフィック量を監視する方法である．この方法では，NIDS ノードのトラフィック量が一定値を超えてからトラフィック量が調整されるまでに，NIDS ノードからソフトウェアロードバンサに通知する手続きが必要である．監視方法 2 はソフトウェアロードバランサが各 NIDS ノードに割り当てているトラフィック量を監視する方法である．この方法では，トラフィック量が一定値を超えた NIDS ノードに対して，即座に割り当てているトラフィック量の調整が可能である．

前述の要件を考慮すると，監視方法 1 は通知の手続きによって負荷情報が反映されるまでの時間が長くなる．このため，監視方法 2 の方がより安定した負荷分散を実現できると考えられるため，本研究では監視方法 2 を負荷分散に採用する．

2.4 性　　能

前述のように，ソフトウェアロードバランサは最小限のパケットロスで高いスループットのトラフィックを処理できるアーキテクチャが必要である．2.2 節で採用したトラフィック分割手法において，ソフトウェアロードバランサがトラフィックを分割するために必要な処理は，パケット転送とパケットの宛先書き換え処理である．以下，最適なアーキテクチャの設計方針を検討する．

まず，ソフトウェアロードバランサのパケット転送は，レイテンシに寛容でパケットロスに厳しいパケット処理と考えられる．このような処理の例として広帯域ネットワーク向けのネットワークアナライザが挙げられる．広帯域ネットワーク向けのネットワークアナライザでは，受信したパケットを libpcap-mmap[16] や PF RING[17] などのように大容量のバッファに格納した後に処理を行うことが一般的である．大容量のバッファを持つことで，アナライザが処理可能なスループットを超えるトラフィックに対しても，バッファが許容できる容量であればパケットを破棄せずに一時的に保持することができる．ソフトウェアロードバ

Vol.2011-OS-119 No.3
Vol.2011-EMB-23 No.3
2011/11/29

ランサの処理においても，広帯域ネットワーク向けのネットワークアナライザと同様に考えることができるため，本研究では受信したパケットを一時的に保持する大容量のバッファを採用する．

次に，ソフトウェアロードバランサのパケット転送処理およびパケットの宛先書き換え処理は，高いスループットを達成するために CPU やメモリを効率よく利用する必要がある．今日の多くの汎用 PC がマルチコア CPU を採用しており，シングルスレッドの処理と比較してマルチスレッドによる並列処理がより効率的にマルチコア CPU を利用可能である．ソフトウェアロードバランサの各処理はパケット毎に独立であるため，パケット毎に処理を並列化することができる．一方で，処理の並列化によって，シングルスレッドによる処理よりもジッタが増加することが考えられるが，前節で述べたように並列 NIDS においてはジッタの影響は特別に考慮する必要はない．さらに，各 NIDS ノードはパケットの到着順序が変更された場合でもコネクションの再構築が可能である[9)–11)] ため，ソフトウェアロードバランサにおけるパケットの到着順序保証は特別に考慮することなく並列化できる．このため，ソフトウェアロードバランサの各処理はマルチスレッドによる並列化に適した処理であるといえる．よって，本研究ではソフトウェアロードバランサにおけるパケット転送処理およびパケットの宛先書き換え処理のマルチスレッドによる並列化を採用する．

3. 実　　装

本節では，前節で検討した設計方針に基づくソフトウェアロードバランサの実装に関して述べる．3.1 節で，実装で採用した基盤技術である PF RING と L2NAT について概説し，3.2 節で基盤技術を用いた実装の詳細について説明する．さらに，次節にて実装したソフトウェアロードバランサのパケット転送の性能評価を行う．

3.1 基盤技術

3.1.1 PF RING

PF RING はユーザアプリケーションに対して，パケットキャプチャおよびパケット送信機能を提供するオープンソースのカーネルモジュールである．PF RING はデフォルトで 4096 パケットを保持できる大容量のリングバッファをユーザアプリケーションに提供する．よって，前節で述べたように，レイテンシに寛容でパケットロスに厳しいアプリケーションに適している．PF RING はユーザアプリケーションからの呼び出しによって，受信用または送信用のリングバッファをカーネル空間とユーザ空間の共有メモリ上に生成する．ユーザアプリケーションは生成されたリングバッファを介して，ネットワークデバイスに対するパケットの受信または送信を行う．

PF RING はネットワークデバイスとユーザアプリケーション間でのパケット送受信をサポートするため，ソフトウェアロードバランサにおけるパケット転送およびパケットの宛先書き換え処理をユーザアプリケーションとして実装できる．これにより，これらの処理はマルチスレッドプログラムとして実装することで容易に並列化できる．前節で述べたように，パケット転送およびパケットの宛先書き換え処理は並列化に適しており，今日の汎用 PC の特徴であるマルチコア CPU を効率的に利用することでスループットの大幅な改善が見込める．

一方，libpcap などの一般的なパケットキャプチャでは，システムコールによってネットワークデバイスからパケットを取得するが，高いスループットのトラフィックに対してはシステムコールがボトルネックになることが知られている．PF RING ではシステムコールではなく，共有メモリ上のリングバッファを介してネットワークデバイスからパケットを取得することでスループットを大幅に改善している．さらに，PF RING では同様の仕組みによるユーザアプリケーションからのパケット送信機能もサポートしており，ユーザアプリケーションでパケット転送が実現できるという特徴がある．共有メモリを介する高スループットのパケットキャプチャとして PF RING の他に libpcap-mmap が挙げられる．しかし，libpcap-mmap がサポートするのはパケットキャプチャの機能に限られるため，パケット転送には PF RING が適しているといえる．以上の特徴から，本研究ではソフトウェアロードバランサのパケット転送に PF RING を利用する．

3.1.2 L2NAT

前節で述べたように，提案するソフトウェアロードバランサではパケットの宛先を書き換えることでトラフィックを分割する．パケットの宛先を書き換える際，侵入検知の機能を保つために各 NIDS ノードの解析に影響を与えるパケットデータは保持する必要がある．一般的に，NIDS ソフトウェアで解析に利用するパケットデータは IP 層およびその上位層のヘッダとペイロードである．これより，ソフトウェアロードバランサで宛先 MAC アドレスを割り当て先 NIDS ノードの MAC アドレスに書き換えることで，パケットの宛先書き換えによるトラフィック分割手法を実現できる．このような宛先 MAC アドレスの書き換えは一般的に L2NAT (Layer 2 Network Address Translation) と呼ばれる．以上より，本研究ではソフトウェアロードバランサにおけるパケットの宛先書き換え処理に L2NAT を利用する．

Vol.2011-OS-119 No.3
Vol.2011-EMB-23 No.3
2011/11/29

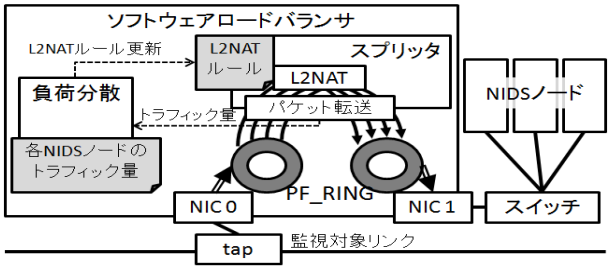

図 4 提案するソフトウェアロードバランサの実装
Fig. 4 The proposed implementation.

3.2 提案するソフトウェアロードバランサの実装

提案するソフトウェアロードバランサの実装を **図 4** に示す．ソフトウェアロードバランサは PF RING，スプリッタ，負荷分散の 3 つのコンポネントから構成される．PF RING は前述のように大容量のリングバッファによってスプリッタとネットワークデバイス間でパケットの受信または送信を行う．スプリッタはパケット転送と L2NAT の機能を組み合わせることで監視トラフィックを分割する．スプリッタはパケットの IP アドレスやポート番号と割り当て先 NIDS ノードの対応を L2NAT ルールとして保持し，L2NAT ルールに従って処理を行う．負荷分散はトラフィックの割り当て先 NIDS ノードとトラフィック量をスプリッタから取得し，各 NIDS ノードに割り当てられているトラフィック量を監視する．特定の NIDS ノードに割り当てられているトラフィック量が一定値を超えた場合には，その NIDS ノードに割り当てられているトラフィックを他の NIDS ノードへ割り当てられるように L2NAT ルールを更新する．提案するソフトウェアロードバランサはスプリッタと負荷分散を含む一つのマルチスレッドプログラムとして実装した．なお，マルチスレッドプログラムにおいて，マルチコア CPU を効率よく利用するために，各スレッドが別々の CPU コアにバインドされるように実装した．以下，スプリッタと負荷分散実装について説明する．

まず，スプリッタにおける処理はマルチスレッドによる並列処理として実装した．各スレッドは，受信ネットワークデバイスから PR RING を用いてパケットデータを取得する．次に，パケットの IP アドレスから L2NAT ルールを検索し，ルールに従って L2NAT でパケットの宛先を書き換える．最後に，パケットを送信ネットワークデバイスの PF RING に転送する．L2NAT はパケットの送信元および送信先 IP アドレスのハッシュ値から該当するルールを検索するハッシュテーブルとして実装した．一般的なテーブルの線形検索ではテーブル数に応じて検索時間が異なるため，安定したスループットで処理することができない．一方，ハッシュテーブルはハッシュ値の計算とテーブル参照という固定的なオーダの処理によって検索が可能であり，ハッシュ値の計算時間が小さい場合は高速なテーブル検索が可能である．さらに，テーブル検索の高速化のためにハッシュ値の生成に計算量が小さい XOR を利用した．加えて，スプリッタではトラフィック量を監視するために，L2NAT のルール毎に処理したパケット数を管理するよう実装した．

次に，負荷分散はマルチスレッドプログラムの 1 スレッドとして実装した．負荷分散スレッドは一定時間毎にスプリッタが管理する L2NAT ルール毎のパケット数を収集し，この情報から各 NIDS ノード毎に割り当てられているトラフィック量を pps 単位で監視する．特定の NIDS ノードのトラフィック量が一定値を超えた場合，この NIDS ノードに割り当てているトラフィックの一部を，トラフィック量が最小の NIDS ノードに割り当てるように L2NAT ルールを変更する．スプリッタと負荷分散を別スレッドとして独立させることで，負荷分散スレッドはスプリッタのスループットを劣化させることなく各 NIDS ノードに割り当てているトラフィック量を調整できる．

4. 評　　価

本節では実装したソフトウェアロードバランサにおけるパケット転送の性能評価を行う．性能評価では，既存研究のソフトウェアロードバランサで利用されているブリッジデバイス[18] を用いたパケット転送の性能と比較し，提案実装におけるパケット転送性能が改善されていることを示す．以下で評価環境を説明し，その後に評価環境におけるソフトウェアロードバランサの性能について評価する．

4.1 評 価 環 境

提案実装におけるパケット転送性能の評価を行うため，**図 5** に示す評価環境を構築した．評価環境は汎用ハードウェアで構成されており，ソフトウェアロードバランサは Intel Xeon X5670 2.93GHz を搭載した汎用 PC で実装し，実験用トラフィックを生成する送信ノードと受信ノードを Intel Xeon W3050 2.53GHz を搭載した汎用 PC で実装した．各 PC の NIC には chelsio N310E を利用し，PC 間は DAC (Direct Attached Cable) および汎用 L2 スイッチによって 10Gbps ネットワークで接続されている．

実験用トラフィック送信ノードはオープンソースの UDP パケット生成ツール pktgen[19] を用いて実装した．pktgen は UDP トラフィックを生成する機能を持つカーネルモジュールである．pktgen ではユーザ空間とカーネル空間のデータ転送のオーバヘッドがないため，

Vol.2011-OS-119 No.3
Vol.2011-EMB-23 No.3
2011/11/29

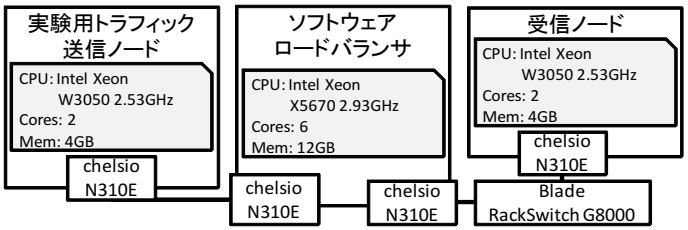

図 5 評価環境
Fig. 5 Experimentation environment.

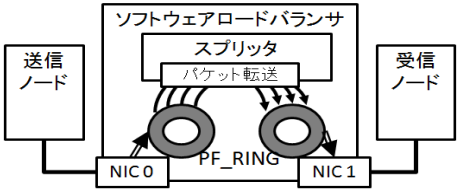

図 6 提案実装におけるパケット転送
Fig. 6 Packet forwarding of the proposed software-based load balancer.

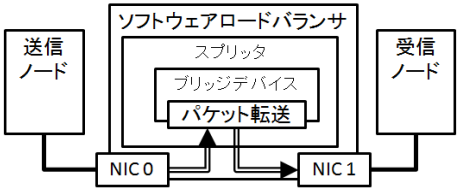

図 7 ブリッジデバイスを用いるパケット転送
Fig. 7 Packet forwarding with bridge device.

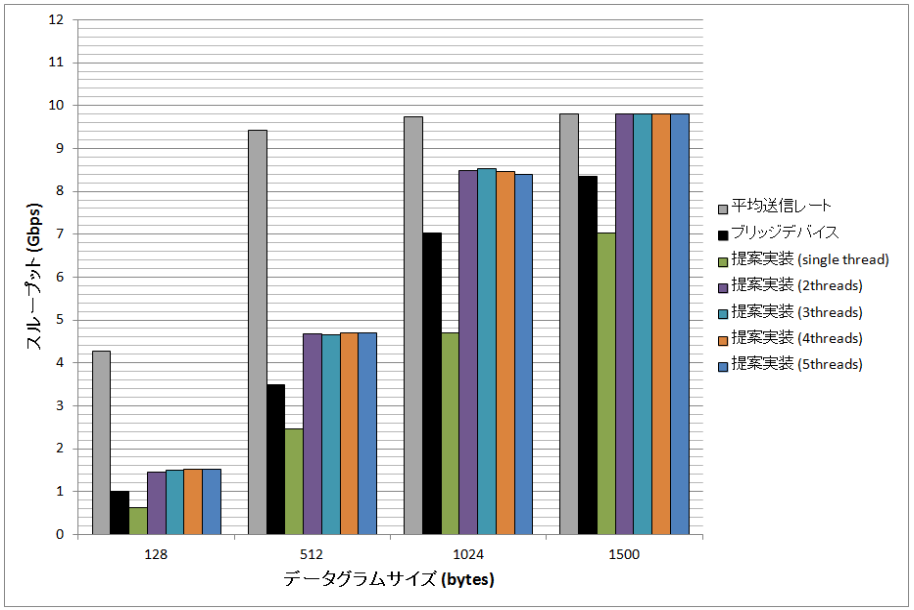

図 8 パケット転送のスループット (Gbps)
Fig. 8 Throughput of packet forwarding (Gbps).

ユーザアプリケーションのトラフィック生成ツールよりも高いスループットのトラフィックを生成することができる．

提案実装におけるパケット転送，および，既存研究で用いられているブリッジデバイスによるパケット転送を**図 6**，**図 7** に示す．提案実装におけるパケット転送は前節で実装したソフトウェアロードバランサのスプリッタにおいて，L2NAT の機能を除いた処理である．ブリッジデバイスによるパケット転送は文献[14] で提案されている汎用ハードウェアベースのロードバランサのパケット転送に利用されている．ブリッジデバイスは複数のネットワークデバイス間でパケットを転送するために一般的に用いられる仮想ネットワークデバイスである．

4.2 パケット転送性能

前述の評価環境によって，ソフトウェアロードバランサのパケット転送におけるスループットとパケットロスを評価する．送信ノードで実験用トラフィックとして固定サイズの 100M 個のデータグラムから構成される UDP トラフィックを生成し，提案実装におけるパケット転送およびブリッジデバイスを用いるパケット転送のスループットとパケットロス率を計測した．計測において，提案実装におけるパケット転送はマルチスレッドプログラムであることから，パケット転送処理のスレッド数によってスループットおよびパケットロスが異なる可能性がある．加えて，提案実装における各スレッドは各 CPU コアにバインドされること，および，負荷分散に 1 スレッド利用すること，さらには提案実装に利用する CPU が 6 コアであることから，スレッド数が 1 から 5 の場合で評価実験を行った．

まず，提案実装におけるパケット転送，および，ブリッジデバイスを用いるパケット転送のスループット (Gbps) を**図 8** に示す．図 8 より，ブリッジデバイスを用いるパケット転送では，1500 バイトのトラフィックにおいても 8.5Gbps 程度のスループットが限界である．よって，既存研究で提案されているソフトウェアロードバランサでは，10Gbps ネットワークにおいて，ブリッジによるパケット転送がボトルネックになると考えられる．一方，提案実装におけるパケット転送ではスレッド数が 2 以上の場合に，どのデータグラムサイズにお

Vol.2011-OS-119 No.3
Vol.2011-EMB-23 No.3
2011/11/29

表 1 パケット転送の最大スループット (Kpps)
Table 1 Maximum throughput of packet forwarding (Kpps).

	スループット (Kpps)
ブリッジデバイス	974.5221963
提案実装 (1 thread)	615.0915687
提案実装 (2 threads)	1412.303957
提案実装 (3 threads)	1467.628956
提案実装 (4 threads)	1481.074426
提案実装 (5 threads)	1477.232266

表 2 パケット転送におけるパケットロス率
Table 2 Packet loss rate of packet forwarding.

	パケットロス率 (%)
ブリッジデバイス	14.592555
提案実装 (1 thread)	28.26735
提案実装 (2 threads)	0
提案実装 (3 threads)	0
提案実装 (4 threads)	0
提案実装 (5 threads)	0

いてもブリッジデバイスによるパケット転送よりも高いスループットを達成しており，1500 バイトのトラフィックに対しては送信レートと等しい 9.8Gbps を超えるスループットを達成した．

次に，提案実装におけるパケット転送，および，ブリッジデバイスを用いるパケット転送の最大スループット (Kpps) を**表 1** に示す．pps を基準とするスループットは提案実装におけるパケット転送，ブリッジデバイスを用いるパケット転送のどちらの場合においても 128 バイトのデータグラムサイズで最大であった．表 1 から，提案実装におけるパケット転送は約 1.5Mpps を達成した．さらに，提案実装におけるパケット転送はブリッジを用いるパケット転送と比較して，スループットが 1.5 倍程度改善されていることが分かる．

前述の実験において，1500 バイトの UDP トラフィックを生成した際の提案実装におけるパケット転送，および，ブリッジデバイスを用いるパケット転送のパケットロス率を**表 2** に示す．提案実装におけるパケット転送では，スレッド数が 2 以上の時にパケットロスなく約 10Gbps のトラフィックを処理できていることが分かる．さらに，ブリッジデバイスを用いるパケット転送と比較して，提案実装におけるパケット転送ではパケットロス率を大幅に軽減できていることが分かる．

以上の結果から，提案するソフトウェアロードバランサのパケット転送が，既存研究で利用されているブリッジデバイスを用いるパケット転送と比較して性能を改善できていることを示した．しかし，データグラムサイズの小さなトラフィックでは 10Gbps を達成できず，パケットロス率が高いことからソフトウェアロードバランサのパケット転送には性能改善が必要である．PF RING では，PF RING 向けの高速なネットワークデバイスドライバを提供しているが，本論文で利用した chelsio N310E で利用できるドライバは提供されていない．これより，実装に利用するネットワークデバイスを変更することでパケット転送性能が改善する見込みがある．

5. まとめと今後の課題

本論文では，10Gbps ネットワークにおいてアプリケーションレイヤでの解析を行う汎用ハードウェアベースの並列 NIDS を実現するために，並列 NIDS に向けたソフトウェアロードバランサを設計，実装した．具体的には，提案するソフトウェアロードバランサでは高速なパケットキャプチャおよび転送モジュールである PF RING を利用し，マルチスレッドでパケット転送処理を並列化する高速なパケット転送を実装した．さらに，本論文の評価実験によって，提案実装におけるパケット転送が 9.8Gbps を超えるスループットを達成し，既存研究のソフトウェアロードバランサで採用されているパケット転送よりも高い性能を持つことを示した．

今後，提案するソフトウェアロードバランサにおけるパケット転送が小さなデータグラムサイズのトラフィックに対して十分な性能を達成するために，ネットワークデバイスの変更による性能改善を検討する予定である．これに加えて，実装したロードバランサの L2NAT 機能を含むトラフィック分割処理の性能，および，負荷分散を評価する予定である．

参 考 文 献

1) Northcutt, S. and Novak, J.: *Network Intrusion Detection An Analyst's Handbook*, Sams Publishing (2002).
2) Han, S., Jang, K., Par, K. and Moon, S.: PacketShader: A GPU-Accelerated Software Router, *ACM SIGCOMM Computer Communication Review*, Vol. 40, No. 4, pp.192–206 (2010).
3) Schaelicke, L. and Freeland, C.: Characterizing Sources and Remedies for Packet Loss in Network Intrusion Detection Systems, *Workload Characterization Symposium, Proceeding of the IEEE International*, pp.188–196 (2005).
4) Hall, M. and Wiley, K.: Capacity Verification for High-Speed Network Intrusion Detection Systems, *Proceedings of the 5th International Conference on Recent Advances in Intrusion Detection*, pp.239–251 (2002).
5) Roesch, M.: Snort - Lightweight Intrusion Detection for Network, *in Proceedings of the 13th USENIX conference on System administration*, pp.229–238 (1999).
6) Paxon, V.: Bro: A System for Detecting Network Intruders in Real-Time, *Computer Networks: The International Journal of Computer and Telecommunications Networking*, Vol.31, No.23–24, pp.2435–2463 (1999).
7) Clark, C., Lee, W., Contis, D., Kone, M. and Thomas, A.: A Hardware Platform for

Vol.2011-OS-119 No.3
Vol.2011-EMB-23 No.3
2011/11/29

Network Intrusion Detection and Prevention, *In Proceedings of the 3 rd Workshop on Network Processors and Applications*, pp.99–118 (2004).
8) Lee, J., Hwang, S. and Park, N.: A High Performance NIDS Using FPGA-based Regular Expression Matching, *Proceedings of The 2007 ACM Symposium on Applied Computing*, pp.1187–1191 (2007).
9) Handley, M., Kreibich, C. and Paxon, V.: Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics, *Proceedings of the 10th conference on USENIX Security Symposium*, Vol.10, pp.9–26 (2001).
10) Sommer, R. and Paxon, V.: Enhancing Byte-Level Network Intrusion Detection Signatures with Context, *10th ACM Conference on Computing and Communication Security*, pp.262–271 (2003).
11) H., D., Feldmann, A., Mai, M., Paxon, V. and Sommer, R.: Dynamic Application-Layer Protocol Analysis for Network Intrusion Detection, *Proceedings of the 15th conference on USENIX Security Symposium*, Vol.15, pp.257–272 (2006).
12) Schaelicke, L., Wheeler, K. and Freeland, C.: SPANIDS: A Scalable Network Intrusion Detection Loadbalancer, *Proceeding of The 2nd Confrerence on Computing Frontiers*, pp.315–322 (2005).
13) Vallentin, M., Sommer, R., Lee, J. amd Leres, C., Paxon, V. and Tierney, B.: The NIDS Cluster: Scalable, Stateful, Network Intrusion Detection on Commodity Hardware, *In Proceedings of the International Symposium on Recent Advances in Intrusion Detection*, pp.107–126 (2007).
14) Krugel, C., Valeur, F., Vigna, G. and Kemmerer, R.: Stateful Intrusion Detection for High-Speed Networks, *Proceedings of The 2002 IEEE Symposium on Security and Privacy*, pp.285–293 (2002).
15) Hanaoka, M., Kono, K., Tirotsu, T. and Abe, H.: Performance Inmprovement by Coordinating Configurations of Independantly-Managed NIDS, *IJCSNS International Journal of Computer Science and Network Security*, Vol.11, No.5, pp.1–11 (2011).
16) Phi Wood: L14:44 2011/10/26ibpcap, `http://public.lanl.gov/cpw/`.
17) Ntop: PF Ring: High-Speed Packet Capture, Filtering and Analysis, `http://www.ntop.org/products/pf_ring/`.
18) The Linux Foundation: Bridge, `http://www.linuxfoundation.org/collaborate/workgroups/networking/bridge`.
19) TSlab: Pktgen: Open Source Traffic Analyzer, `http://tslab.ssvl.kth.se/pktgen/`.