論　文

仮想NICへのパケット処理オフロードによる仮想スイッチ処理の負荷低減手法の提案

川島　龍太[†a)]　　松尾　啓志[†]

Virtual NIC Offloading Approach for Improving Performance of Virtual Networks

Ryota KAWASHIMA[†a)] and Hiroshi MATSUO[†]

あらまし　データセンターにおいて仮想スイッチとIPトンネリングを用いて仮想ネットワークを構築するエッジ・オーバレイ方式が注目されている．しかし，仮想スイッチはソフトウェア実装のため，負荷の大きいトンネリングや暗号化処理によって仮想ネットワーク全体の性能が低下するという問題がある．これまで仮想スイッチ処理をハードウェアにオフロードして負荷を低減させる手法が提案されているが，機能追加などの柔軟性や既存システムとの適合性の問題が新たに生じてしまう．本研究では，トンネリングや暗号化及びOpenFlowによるパケット変換処理をVMの仮想NICにオフロードすることで仮想スイッチの負荷を低減させる手法を提案する．提案手法では，上記のパケット処理を仮想スイッチではなく複数のVM上で並列に行うことで仮想ネットワークの性能を改善する．本論文では，提案手法の設計及び実装について述べるほか，仮想NIC機能を設定するための制御プロトコルについても言及する．性能評価を行った結果，VM間通信のフロー数が増えるに従って，提案手法を用いない場合は総スループットが低下したが，提案手法を用いた場合は総スループットが向上した．

キーワード　Software-Defined Networking，仮想ネットワーク，仮想NIC，OpenFlow，エッジ・オーバレイ

1. まえがき

パブリッククラウドサービスを提供するデータセンターでは，マルチテナント機能を実現するために，各テナントが独立性のある論理ネットワーク（仮想ネットワーク）を使用する必要がある．一般的に，仮想ネットワークは，テナントがもつ複数の仮想マシン（VM）と仮想ファイアウォールなどの仮想的な機器によって構成され，それぞれの機器が論理的なリンクによって相互に接続されている．各テナントが利用する仮想ネットワークは，実際には同じ物理リソースを共有しているため，物理—仮想間の橋渡しを行う仮想スイッチと，各仮想ネットワークのトラヒックを論理的に分離する仕組みが必要になる．

最近では，仮想スイッチとIPトンネリングを用いて仮想ネットワークを構築するエッジ・オーバレイ方式（または分散トンネリング方式）[1]が注目されている．具体的には，各物理サーバ上の仮想スイッチ間でVXLAN [2]などのIPトンネル（L2-in-L3）を構築し，各仮想ネットワークのパケットに付加された固有のID（トンネルID）によってトラヒックを区別する．また，仮想スイッチにおけるフレーム転送をOpenFlow [3]によって柔軟に制御する仕組みも普及しつつあり，仮想スイッチの重要性は今後ますます高まっていくと考えられる．

ところが，通常の仮想スイッチはハードウェアではなくソフトウェアコンポーネントとして実装されているため，高速な物理ネットワークと多数のVMを有するデータセンターにおいては，ソフトウェアによるパケット処理（特にトンネリングなどの負荷の大きい処理）が仮想ネットワーク全体の性能に影響を与えてしまう．したがって，物理サーバ上で行われる，ソフトウェアによるパケット変換処理を効率化し，仮想ネットワークの性能低下を防ぐ仕組みが新たに必要になる．

これまで，ハードウェア支援技術を用いて，仮想スイッチの処理負担を低減させる手法が提案されてきた．EVB (Edge Virtual Bridging) はIEEE 802.1Qbg [4]として標準化されている技術で，仮想スイッチのフレー

† 名古屋工業大学情報工学科，名古屋市
Dept. of Computer Science, Nagoya Institute of Technology, Gokiso, Showa-ku, Nagoya-shi, 466–8555 Japan
a) E-mail: kawa1983@nitech.ac.jp

ム転送処理を物理スイッチにオフロードするVEPA方式と，物理NICにオフロードするVEB方式がある．また，Luoらは，プログラマブルなNICを用いて物理NIC上でのOpenFlow処理を実現している[5]．

このようなハードウェア支援技術は，仮想ネットワーク性能の改善という点では効果的であるが，全ての機能をハードウェアとして実現すると，仮想的なネットワーク機器がもたらす，メンテナンスの容易性，機能追加の容易性及び機器更新の低コスト性といった利点[6]を享受することができなくなる．また，OpenFlowやVXLANなど，仕様が比較的頻繁に更新されるプロトコルを利用する場合，仮想的なネットワーク機器であればソフトウェアの更新で対応できるが，ハードウェア機器の更新は，データセンター内でのライフサイクル（通常3–5年[7]）に合わせて行うことが基本となるため，仕様変更への迅速な対応を毎回行うことは困難である(注1)．

そこで本研究では，ハードウェア支援技術に依存しない，ソフトウェア的な手法によって，仮想ネットワーク性能を改善するシステムを提案する．提案システムでは，OpenFlowのMatch/Action処理，VXLANなどのトンネリング処理，暗号化処理などの高度なパケット処理をVMの仮想NIC内で実行するための仕組みであるCVSW (Co-Virtual SWitch)を開発した．加えて，各VMの仮想NIC内にあるCVSWを，リモートから一元管理するためのコントローラと制御プロトコルも実現した．

提案システムによって，従来では仮想スイッチが行っていた高度なパケット処理を，送信元/受信先のVMが実行するようになるため，仮想スイッチの負荷が低減し，仮想ネットワーク全体の性能が向上する．特定のVM間通信において負荷の大きいパケット処理が必要となる場合でも，他のVM間の通信には影響を及ぼさないという利点もある．一般に，Open vSwitch [8]などの代表的な仮想スイッチは，データプレーン用のパケット処理（Fast Path）をシングルスレッドで行っている[9]ため，CVSWはこれらを暗黙的に並列処理する手法であると言える．また，CVSWによるパケット処理は，VMの仮想NIC内に閉じているため，既存の仮想スイッチをそのまま利用することができ，更には上記のハードウェア支援技術や提案手法を用いないVMとの共存も可能である．

(注1)：ファームウェアアップデートで対応できない場合．

提案システムでは，仮想NIC内での実装というCVSWの特徴を利用して，MTU (Maximum Transfer Unit)サイズといった，仮想NICのパラメタ値を動的に設定する仕組みも実現した．エッジ・オーバレイ方式では，トンネリングのためのカプセル化処理によって，パケットサイズが物理ネットワークのMTUサイズを超過し，IPフラグメンテーションが発生する（性能が低下する）ことが問題となっている[1]．そこで，提案システムでは，コントローラが最適なMTUサイズを各CVSWに通知し，仮想NICのMTUサイズを直接変更することで，この問題を回避している．

本論文では，提案システムの構成と，Linuxの準仮想化ネットワークドライバを用いたCVSWの実装及びFloodlight [10]を用いたコントローラの実装について説明する．また，実装したシステムを用いて行った性能評価実験についても紹介する．評価の結果，VM間通信のフロー数が増えるに従って，提案手法を用いない場合は総スループットが低下したが，提案手法を用いた場合は総スループットが向上した．

以下，**2.**では関連研究について紹介する．**3.**では，関連研究について述べた課題を踏まえ，提案システムが満たすべき要件について言及する．**4.**では，提案システムの構成とCVSWの制御プロトコルについて，**5.**では提案システムの内部実装について説明する．続いて**6.**では，提案システムの性能評価実験について述べる．最後に**7.**で本論文のまとめと今後の課題について言及する．

2. 関連研究

EVBはIEEE 802.1Qbgとして標準化されている技術で，仮想スイッチのフレーム転送処理をハードウェアにオフロードすることで負荷を低減させる．具体的には，物理サーバが直接接続している物理スイッチで転送処理やフィルタリング処理を行うVEPA（Virtual Ethernet Port Aggregator）方式と，同様の処理を物理NICで行うVEB（Virtual Ethernet Bridging）方式が存在する．しかし，同一サーバ上でのVM間通信では逆に性能が低下する[11]とされているほか，より高度なパケット処理に関してはベンダ依存となる．

nSwitching [12]は同一サーバ上でのVM間通信に特化した，物理NIC内に実装されたハードウェア支援システムであり，仮想マシンと物理NIC間で直接パケットを送受信できるSR-IOV (Single Root I/O Virtualization) [13]技術を応用して効率的なVM間

通信を実現している．nSwitching では，SR-IOV によってフレーム転送における物理サーバの CPU 使用量を削減しているほか，アクセス制御リスト（ACL），QoS (802.1p)，モニタリングなど付加機能も提供されている．しかし，SR-IOV の利用によって仮想スイッチ処理がバイパスされてしまうため，OpenFlow や IP トンネルを用いたエッジ・オーバレイ方式への対応は困難である．

Luo ら [5], [14] は，ネットワークプロセッサを用いて，OpenFlow 処理が可能な物理 NIC を実現している．ただし，Slow Path と呼ばれる，コントローラ連携などの処理は物理 NIC ではなく，物理サーバ内の Open vSwitch が行っているため，物理 NIC との間で，制御コマンドをやり取りするための仕組みを Open vSwitch 内に追加する必要がある．その結果，特定のハードウェア機器に依存しない，機能追加がハードウェア機器と比較して容易という，Open vSwitch の利点が損なわれてしまう．

vNFC [15] は，仮想スイッチが対応していないアプリケーション層の機能（e.g. アプリケーションファイアウォール）を，個々の VM に対して動的かつ透過的に適用するためのソフトウェアコンポーネントである．vNFC は動的リンクライブラリとして実装されており，VM の起動時に QEMU [16] へ動的にリンクされる．vNFC の機能は，OpenFlow コントローラから，ベンダ拡張メッセージによって制御されている．vNFC は提案手法と同様に，VM ごとに実体をもち，並列に動作することが可能であるが，仮想スイッチの処理をオフロードすることは想定されていない．また，SR-IOV や SV-NIC [17] など，VMM をバイパスする技術と併用することができず（vNFC は VMM に動的リンクするため），仮想—物理間で高速なパケット転送処理を行うことはできない，VM マイグレーションの際には再設定が必要，などの課題がある．

3. 提案システムの要件

本章では，**2.** で述べた関連研究が抱える課題を踏まえて，提案システムが満たすべきシステム要件を定義する．表 1 に，提案システムの要件一覧と関連研究での対応状況を示す．

まず機能面について，提案システムは，OpenFlow 及びエッジ・オーバレイ方式に基づいたネットワーク環境での利用を想定しているため，仮想スイッチが本来行う OpenFlow 処理やトンネリング処理，IPsec との併用 [2] のための暗号化機能に対応する必要がある．また，カプセル化によるフラグメンテーション問題を回避するため，VM の MTU サイズを調整する機能があると望ましい．EVB と nSwitching は，仮想スイッチによる通常のスイッチング処理をハードウェアオフロードする仕組みのため，上記の機能には対応していない．Luo らのシステムは，OpenFlow の Fast Path 処理を物理 NIC へオフロードするが，トンネリングなどの処理には対応していない．vNFC はユーザ定義のパケット処理機能を実現する仕組みであるため，仮想スイッチが対応している OpenFlow やトンネリングなどの機能は対象としないが，文献 [15] では，専用のコントローラが MTU サイズ超過を示す ICMP メッセージを生成することによって，VM が送出するパケットのサイズを調整する仕組みが提示されている．

表 1　システム要件一覧
Table 1　System requirements.

	要件	提案手法	EVB	nSwitching	Luo et al.	vNFC
機能	OpenFlow (注2)	✓	-	-	✓	-
	トンネリング	✓	-	-	-	-
	MTU 調整	✓	-	-	-	✓
	暗号化	✓	-	-	-	-
非機能	仮想 SW の負荷低減	✓	✓	✓	✓	-
	サーバ内通信性能	✓	-	✓	-	✓
	ハードウェア独立性	✓	-	-	-	✓

続いて非機能要件について見ると，vNFC 以外の手法は，仮想スイッチによるパケット処理をオフロードすることを目的としており，仮想スイッチの処理負荷を低減することが可能である．その中で，提案手法は OpenFlow やトンネリング，暗号化などの高度なパケット処理をオフロードする点が特徴である．ハードウェアオフロードを行う場合は，先述したように，同一物理サーバ上での VM 間通信の性能が低下する問題が生じるが，提案手法と vNFC ではスイッチング処理を仮想スイッチで行うため，このような問題は生じない．最後にハードウェア独立性について述べると，提案手法と vNFC は完全にソフトウェア実装であり，共に VM/VMM 内で動作するため，特定のハードウェア機器には依存しない．エッジ・オーバレイ方式の利点は，既存の物理ネットワーク環境を利用できる点にあるため，特定のハードウェア環境に依存しないことは提案システムにとって必須の要件となる．

(注2)：Fast Path 処理のみで，Slow Path 処理は除く．

4. 提案システムの設計

本章では，提案システムのアーキテクチャ構成，CVSW が実現する SDN（Software-Defined Networking）機能，及びコントローラによる CVSW の制御方法について説明する．ここで，SDN 機能とは，コントローラからの制御を前提とした，パケット処理に関するプログラマビリティ機能及びパラメタ設定機能を指している．

図 1 は提案システムの全体アーキテクチャを表している．提案システムでは，上記の SDN 機能は各 VM が使用する仮想 NIC ドライバ (CVSW) 内に実装されており，ドライバ上を流れる入出力 Ethernet フレームに容易にアクセスすることが可能である．それゆえ，例えば OpenFlow に基づいたパケット変換処理，VLAN タグの付け外し，更には VXLAN などのトンネリング処理も仮想 NIC 内で実行できる．

表 2 は，CVSW によって実行可能な OpenFlow の Match/Action 処理を示している．まず Match 処理に関しては，仮想スイッチと同様に Ethernet フレームを直接扱うため，OpenFlow が対応しているあらゆるプロトコルのヘッダに対して Match 処理を適用できる．しかし，入力（Ingress）ポート番号など，フレーム内に値が含まれない場合は，Match 処理は適用できない．Action 処理も同様に，様々なヘッダフィールドの変更や VLAN タグの付け外しは可能であるが，CVSW にはポートの概念が存在せず，パケットの出力先が固定(注3)であるため，OpenFlow の `OUTPUT` で行うような，出力先ポートの動的決定やコントローラへの転送処理を行うことはできない．また，仮想 NIC ドライバの構造上，QoS 制御（レート制御）のためのキューイング処理 (Set-Queue) も実現困難である．そのため，OUTPUT 処理や Set-Queue 処理は，従来通り仮想スイッチが実施する必要がある．

OpenFlow に基づいたパケット処理のほかに，CVSW は，MTU サイズやハードウェアオフロード設定など仮想 NIC のさまざまなパラメタ設定をコントローラから制御する仕組みも提供している．仮想 NIC の MTU サイズは，物理ネットワークで IP トンネルを使用する場合，IP 層でのフラグメント処理を回避するために値を調整する場合がある．従来では，VM の管理者が手動で MTU サイズを調整するか，NVGRE [18] のように，ICMP を用いて定期的に値を設定する方法しかなかったが，提案システムでは，コントローラが各 CVSW を一元的に管理するため，数万台規模におよぶ仮想 NIC の MTU 値を直接かつ動的に設定することができる．オフロード設定については，現在では多くの OS が LSO（Large Send Offload）[19] や

(注3)：送信パケットは仮想スイッチへ，受信パケットは VM カーネル内の TCP/IP 層へ転送する．

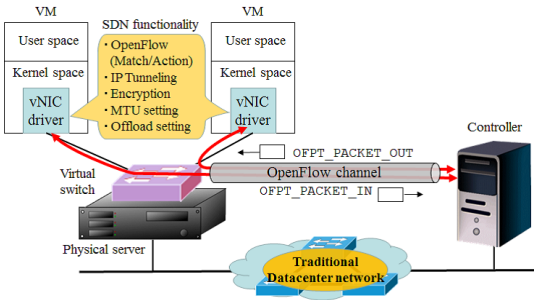

図 1 提案システムの全体アーキテクチャ
Fig. 1 Architectural overview of the proposed system.

表 2 CVSW が対応している OpenFlow 1.0 の Match/Action ルール
Table 2 Supported OpenFlow 1.0 Match/Action rules.

Match ルール	対応
Ingress port	N/A
Ethernet source address	✓
Ethernet destination address	✓
Ethernet type	✓
VLAN ID	✓
VLAN priority	✓
IP source address	✓
IP destination address	✓
IP protocol	✓
IP ToS bits	✓
Transport source port	✓
Transport destination port	✓
ICMP type	✓
ICMP code	✓

Action ルール	対応
Forward the frame to the specified port	N/A
Enqueue the frame (Optional)	N/A
Drop the frame	✓
Set VLAN ID	✓
Set VLAN priority	✓
Strip VLAN header	✓
Modify source MAC address	✓
Modify destination MAC address	✓
Modify IPv4 source address	✓
Modify IPv4 destination address	✓
Modify IPv4 ToS bits	✓
Modify transport source port	✓
Modify transport destination port	✓

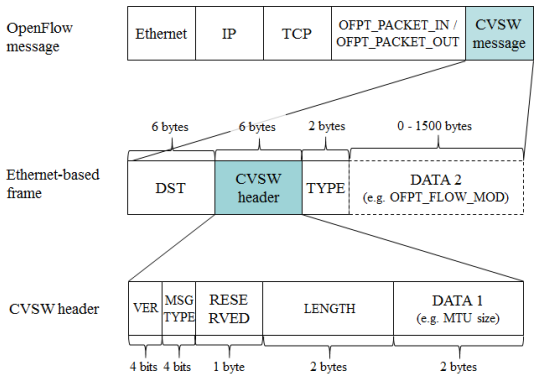

図 2　CVSW 制御メッセージの構造
Fig. 2　An Ethernet-based CVSW message structure.

LRO（Large Receive Offload）[20] などの機能に対応しており，またデフォルトで有効となっている場合もある．しかし，一部の仮想化環境では，これらの設定が有効になっていることで逆に性能が低下するケースもある [21] ため，環境に応じて柔軟に設定を変更できる仕組みが必要である．

4.1　CVSW 制御メッセージ

前述のとおり，CVSW の SDN 機能は，リモートにあるコントローラから一元的に管理される．ただし，各 CVSW は直接コントローラと通信を行うのではなく，接続している仮想スイッチを経由して，間接的にコントローラとメッセージ（CVSW 制御メッセージ）の送受信を行う．ここで，仮想スイッチは OpenFlow に対応しており，コントローラとの間で OpenFlow コネクションを事前に確立しているものとする．仮想スイッチが中継役となることで，既存の OpenFlow の枠組みを利用して CVSW 制御メッセージをやり取りできるようになる．また，各 CVSW がコントローラと通信コネクションを直接確立する必要がないため，CVSW 及びコントローラの実装が容易になるという利点もある．

図 2 は，CVSW 制御メッセージの構造と OpenFlow プロトコルとの関係性を表している．コントローラから見た場合，全ての CVSW 制御メッセージは，OpenFlow 標準の `OFPT_PACKET_IN` あるいは `OFPT_PACKET_OUT` メッセージのペイロードとして送受信される（図上部）．これらの OpenFlow メッセージは，ヘッダを含むフレーム全体をペイロードとして扱うため，仮想スイッチは `OFPT_PACKET_OUT` のペイロードをそのまま Ethernet フレームとして任意のポートに出力することが可能である．この仕組みを利用するため，CVSW 制御メッセージは Ethernet フレームに似た構造（図中部）をしており，`DST` と `TYPE` は本来のものと同じ役割を果たす．Ethernet ヘッダの送信元 MAC アドレス（`SRC`）に該当するフィールドは，提案手法においては，CVSW 制御用のヘッダ（図下部）として使用する(注4)．

表 3　CVSW 制御メッセージ一覧
Table 3　CVSW message types.

Message type	Receiver	Data1	Data2
HELLO	Both	-	-
REGISTER	Controller	-	MAC address
SET ENTRY	CVSW	-	OFPT_FLOW_MOD
DELETE ENTRY	CVSW	-	OFPT_FLOW_MOD
LIST ENTRIES	Controller	-	OFPT_FLOW_MOD
SET MTU	CVSW	MTU	-
SET OFFLOAD	CVSW	on/off	-

ここで，CVSW 制御メッセージの種類を表 3 にまとめる．`HELLO` メッセージは，仮想 NIC ドライバあるいはコントローラが CVSW の仕組みに対応していることを通知するために双方が使用する．`REGISTER` メッセージは，CVSW（仮想 NIC）をコントローラに登録するために，`SET/DELETE ENTRY` メッセージは CVSW がもつ OpenFlow のフローテーブルにエントリを設定するために使用する．フローテーブルは，通常の OpenFlow スイッチと同様に，OpenFlow の `OFPT_FLOW_MOD` メッセージを利用して設定する．

5.　実　　装

筆者らは，Linux 標準の準仮想化ドライバと，OpenFlow 用のコントローラフレームワークを用いて提案システムの実装を行った．本章では，主に CVSW とコントローラに焦点をあてて，その内部実装について説明する．

5.1　仮想 NIC ドライバ（CVSW）

CVSW の機能は，Linux カーネルの一部として提供されている準仮想化ドライバ（virtio_net）内に実装した．具体的には，図 3 に示すように，ドライバのフレーム処理フロー内に，OpenFlow に基づいた Match 処理と Action 処理を追加している．フレーム送信時は，既存の `start_xmit` 関数内で，SDN 機能の処理

(注4)：メッセージを受信する仮想 NIC 内では，送信元 MAC アドレスのフィールドは検査されないため，通信に不都合はない．CVSW 用のヘッダを Ethernet ヘッダ内に定義している理由は，将来，仮想スイッチを通過するデータプレーン用のフレームに対して，入出力ポート番号などのメタデータを付加するためである．

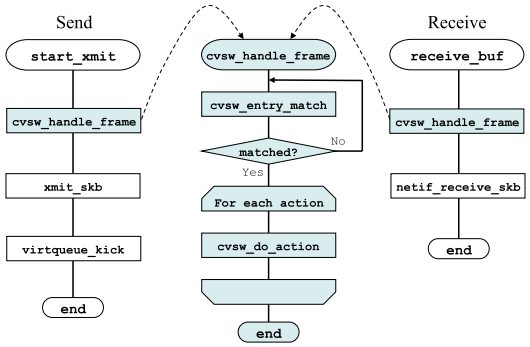

図 3　CVSW (virtio_net) におけるフレーム処理フロー
Fig. 3　A flowchart of frame processing in CVSW.

を行う cvsw_handle_frame 関数を呼び出し，Match 処理と Action 処理を行う．フレーム受信時も同様に，既存の receive_buf 関数内で，受信したフレームを cvsw_handle_frame 関数に渡し，先述の処理を行う．ドライバは，OpenFlow のフローテーブルと同様のフローテーブルを内部で管理しており，これに基づいて Match 処理と Action 処理を行う．

5.2　コントローラ

筆者らは，Floodlight [10] を用いて CVSW 制御用のコントローラを実装した．コントローラは，最初にスイッチが接続すると，CVSW 制御用の HELLO メッセージを送信すると同時に，全ての CVSW 制御メッセージをコントローラへ転送するように，スイッチのフローテーブルを設定する．REGISTER メッセージによって CVSW が登録された後，コントローラは SET_ENTRY メッセージを送信して CVSW のフローテーブルを設定する．また筆者らは，OpenFlow では対応していない，トンネリングの設定を行うために，TUNNEL_ENCAP と TUNNEL_DECAP アクションを新たに定義した．それぞれのアクションは，OFActionTunnelEncap / OFActionTunnelDecap クラスとして Floodlight フレームワークに追加され，設定した値は，OFPT_FLOW_MOD メッセージの一部として仮想 NIC に送信される．

6.　性 能 評 価

本章では，仮想スイッチによるパケット処理を比較対象として，実装した提案システムを利用した際の VM 間通信の性能を評価する．最初に，1 組の VM 間通信を対象に，OpenFlow による基本的なパケット変換処理を行った場合の性能を評価する．続いて，仮想

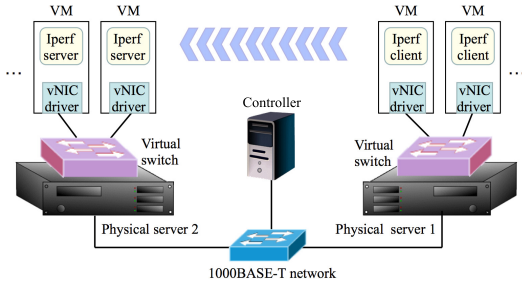

図 4　評 価 環 境
Fig. 4　Experimental environment.

表 4　マシン仕様
Table 4　Machine specifications.

	Physical server 1	Physical server 2	VM
OS	Fedora 18 (3.9.11)		
CPU	Core i7(3.6 GHz)	Core i7(3.4GHz)	1 core
Memory	64 GB	32 GB	2 GB
Buffer	1 MB		
Network	1000BASE-T		-

NIC 設定の最適化が VXLAN 通信に与える影響について，更に複数組の VM 間通信を対象に，負荷の大きいパケット処理を行った場合における並列化の効果を評価する．最後に，**2.** で述べた関連研究を比較対象として，提案システムの性能評価結果について考察を行う．

図 4 と表 4 に，評価で使用したネットワーク環境とマシン仕様を示す．性能の評価は，異なる物理サーバ上で動作している VM 間通信を対象とし，測定には Iperf [22] を用いた．実験では，Iperf クライアントは，Iperf サーバに対して 1 分間連続的に UDP パケットを送信する．また，パケットのサイズは 64，512，2048，4096，32768 bytes の中から一つを選択する．そして，受信側であるサーバの出力結果を評価値として使用する．なお，仮想スイッチ（Open vSwitch 1.10.0）及び CVSW の設定は事前にコントローラが済ませている．

6.1　CVSW における性能オーバヘッドの評価

表 5 は，1 組の VM 間通信を対象にして，付加的なパケット処理を何も行わない（Null processing）ケース，OpenFlow に基づいて，各パケットの IP アドレスを変更するケース，及び VLAN タグの付け外しを行うケースについてそれぞれ測定した結果を示している．なお，“proposed” は提案手法を表し，上記の処理は全て CVSW によって行われ，仮想スイッチは単に物理—仮想 NIC 間でのフレーム転送処理のみを行っている．“existing” の場合は，CVSW の機能は使用

表 5　OpenFlow に基づいたパケット処理性能
Table 5　Throughput of OpenFlow packet processing.

Null processing [Mbps]

	64	512	2048	4096	32768
proposed	19.2	255	925	946	953
existing	19.6	257	925	946	953

IP address (src/dest) modification [Mbps]

	64	512	2048	4096	32768
proposed	19.2	255	926	945	953
existing	19.5	255	925	946	953

VLAN push/pop [Mbps]

	64	512	2048	4096	32768
proposed	19.3	256	922	943	951
existing	19.6	255	922	943	951

表 6　VXLAN スループット
Table 6　Throughput of VXLAN tunneling.

[Mbps]

	64	512	2048	4096	32768
proposed	19.6	258	887	915	924
existing	19.0	256	862	887	200
(adjust)	-	-	876	915	924

せず，仮想スイッチ自身がこれらの処理を全て行っている．

実験結果を見ると，どのパケットサイズにおいても，提案手法と既存手法との間に顕著な差は見られない．また，三つの処理内容の結果がどれも同程度であることから，IP アドレスの変更や VLAN タグの付け外しは軽量な処理であり，VM 間の通信性能には大きな影響を与えないことが分かった．

6.2　仮想 NIC 設定による VXLAN 性能の評価

続いて，コントローラが適切な MTU 値を仮想 NIC に設定することで，VXLAN を使用する仮想ネットワークの性能が向上することを表 6 に示す．“proposed” では，CVSW 制御メッセージによって，仮想 NIC の MTU サイズを 1450 に設定している．参考として，`ifconfig` コマンドによって直接 MTU 値を変更した場合の既存手法の測定結果を，“adjust” として記す．

測定結果を見ると，既存手法を用いた場合は，IP フラグメンテーションが発生する 2048，4096，32768 bytes において，提案手法の性能を明らかに下回っていることがわかる．特にパケットサイズが 32768 bytes の場合は，既存手法の性能が著しく低下しており，VXLAN におけるフラグメント処理がボトルネックになることを示している．提案手法と “adjust” の結果が示すように，仮想 NIC の MTU 値を適切に設定することで，このボトルネックを回避することができる．

表 7　VXLAN 及び AES 暗号化利用時のスループット
Table 7　Throughput of VXLAN and AES encryption.

1 VM [Mbps]

	64	512	2048	4096	32768
proposed	19.5	259	789	825	830
existing	19.3	213	844	834	96.1

2 VMs [Mbps]

	64	512	2048	4096	32768
proposed	19.4	260	441	455	456
existing	19.2	232	391	362	N/A

4 VMs [Mbps]

	64	512	2048	4096	32768
proposed	19.4	198	222	229	228
existing	16.8	143	173	139	N/A

6.3　複数 CVSW による並列処理の評価

最後に，負荷の大きいパケット処理を行う場合に，提案手法によって負荷が分散化される場合の効果を評価する．ここでは，VXLAN と IPsec を使用して仮想ネットワークを実現する状況を想定して，仮想スイッチ及び仮想 NIC 内に，共通の AES 暗号化/復号機能を実装した(注5)．また，両手法とも，仮想 NIC の MTU サイズを事前に調整しており，VXLAN 処理に伴うフラグメンテーションは発生しない．

表 7 は，同時に通信を行う VM の組を 1，2，4 と変化させた場合の，フロー単位での平均スループットを表している．提案手法では仮想 NIC 内で，既存手法では仮想スイッチ内で，フロー中の各パケットに対して，VXLAN 及び AES の処理が行われる．実験結果を見ると，複数組による通信の場合，特にパケットサイズが 512 bytes 以上の場合において，提案手法の性能が既存手法を明らかに上回っていることが分かる．また，パケットサイズが 32768 bytes の場合，既存手法ではほとんど通信を確立できない結果となった．これは，既存手法においては，仮想スイッチにおけるパケット処理速度が，VM からのパケット到着速度に追い付かず，ボトルネックになっていることが原因であると考えられる．

ここで，各 VM 間通信における使用帯域幅を集約したグラフを図 5 に示す．図中の “P” は提案手法を，

(注5)：本評価では，パケットごとの処理負荷に着目するため，IPsec における鍵生成や鍵交換などの機能は実装していない．

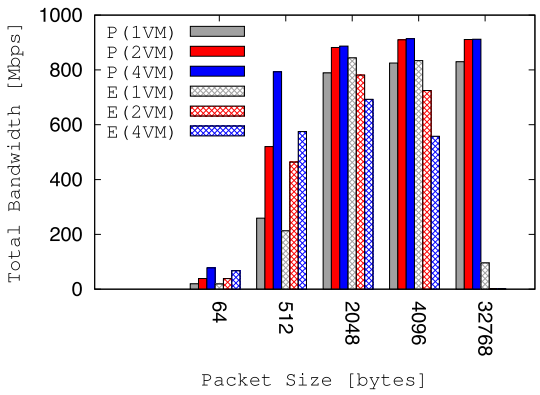

図 5　VXLAN/AES 処理を行った際の総スループット
Fig. 5　Total bandwidth of VM-to-VM links using VXLAN/AES processing.
('P': proposed, 'E': existing)

"E" は既存手法をそれぞれ表している．提案手法では，VM 間通信の組が増えるに従って，トータルの帯域幅が向上しているが，既存手法においては，パケットサイズが 2048 bytes 以上の場合，逆にトータルの帯域幅が減少していることが分かる．

6.4 考　　察

前節までの結果から，提案システムは，暗号化などの負荷が大きい処理をオフロードする場合に性能を向上できることが分かった．そのため提案システムは，エッジ・オーバレイ方式など，高度なパケット処理機能を仮想スイッチへ移行するネットワーク [23] において特に有効であると言える．一方で，代表的なハードウェア支援技術である EVB や，それをベースとする nSwitching は，単純なスイッチング処理を行う仮想スイッチを対象としており，提案手法と比べて高速にスイッチング処理を実行できると考えられるが，トンネリングや暗号化，OpenFlow などのオフロードには対応していない．Luo らのシステムは，OpenFlow の Fast Path 処理をハードウェアオフロードしており，文献 [14] によるとパケットの往復遅延時間を 20%程度削減できる．そのため，提案システムよりも効率的に Fast Path 処理を実行できると思われるが，EVB などと同様に，エッジ・オーバレイに必要なトンネリング（+暗号化）機能には対応していない．vNFC は，提案手法と同じくソフトウェアによって高度なパケット処理を行うシステムであるが，その用途は仮想スイッチが対応しないアプリケーション層機能を実現することにあるため，仮想スイッチの処理はオフロードされない．ただし，理論上は vNFC でも OpenFlow の Fast Path 処理，トンネリング処理，暗号化処理を実行することが可能であり，vNFC が VM 数分だけ並列して動作する点を考慮すると，提案手法と同程度の性能を実現できると考えられる．

7. む　す　び

マルチテナント機能を提供するデータセンターにおいて，仮想スイッチと IP トンネリングを用いて仮想ネットワークを構築するエッジ・オーバレイ方式が注目されている．しかし，仮想スイッチはソフトウェアコンポーネントとして実装されているため，負荷の大きいパケット処理が多発すると性能が大きく低下してしまうという問題がある．

本研究では，ハードウェアオフローディングなどに依存せずに，ソフトウェア的なアプローチによって仮想スイッチの負荷を低減する手法を提案した．具体的には，OpenFlow の Match/Action 処理，VXLAN などのトンネリング処理，暗号化処理などの高度なパケット処理を，VM の仮想 NIC 内で実行するための仕組みである CVSW と，CVSW をリモートから一元管理するためのコントローラ及び制御プロトコルを実現した．また，物理ネットワークの特性に応じて仮想 NIC の設定を調整することで，仮想ネットワークの性能を改善することができる．

今後の課題として，提案手法と SR-IOV などの I/O 仮想化技術を組み合わせて，CVSW による高度なパケット処理とハードウェア支援による高速なパケット転送を両立する手法の確立が挙げられる．

文　　献

[1] T. Narten, E. Gray, D. Black, L. Fang, L. Kreeger, and M. Napierala, "Problem statement: Overlays for network virtualization," Internet Draft, 2013.
[2] M. Mahalingam, D. Dutt, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright, "VXLAN: A framework for overlaying virtualized layer 2 networks over layer 3 networks," Internet draft, 2014.
[3] N. McKeown, T. Andershnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turneron, and H. Balakris, "OpenFlow: Enabling innovation in campus networks," ACM Computer Communication Review, vol.38, no.2, pp.69–74, April 2008.
[4] 802.1Qbg - Edge Virtual Bridging, http://www.ieee802.org/1/pages/802.1bg.html
[5] Y. Luo, E. Murray, and T.L. Ficarra, "Accelerated virtual switching with programmable NICs for scalable data center networking," Second ACM SIG-

COMM Workshop on Virtualized Infrastructure Systems and Architectures (VISA 2010), New Delhi, India, Sept. 2010.
[6] 中尾彰宏，“SDN がもたらす柔軟な将来網の世界，” 信学誌，vol.96, no.12, pp.902–905, Dec. 2013.
[7] C. Garnier, “Data centre life cycle assessment guidelines,” The Green Grid, white paper #45, v2, 2012.
[8] Open vSwitch, http://openvswitch.org/
[9] M. Carbone, G. Catalli, and L. Rizzo, “Improving the performance of open vSwitch,” EuroBSDcon 2011, Maarssen, The Netherlands, Oct. 2011.
[10] Floodlight OpenFlow controller - Project Floodlight, http://www.projectfloodlight.org/floodlight/
[11] J. Pettit, J. Gross, B. Pfaff, M. Casado, and S. Crosby, “Virtual switching in an era of advanced edges,” Proc. 2nd Workshop on Data Center — Converged and Virtual Ethernet Switching (DC-CAVES), ITC 22, Sept. 2010.
[12] J. Bardgett and C. Zou, “nSwitching: Virtual machine aware relay hardware switching to improve intra-NIC virtual machine traffic,” Proc. International Conference on Communications (ICC), pp.2700–2705, Ottawa, CA, June 2012.
[13] Single Root I/O Virtualization, http://www.pcisig.com/specifications/iov/single_root/
[14] Y. Luo, P. Cascon, E. Murray, and J. Ortega, “Accelerating OpenFlow switching with network processors,” Proc. 5th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS 09), pp.70–71, 2009.
[15] R. Kawashima, “vNFC: A virtual networking function container for SDN-enabled virtual networks,” Proc. Second Symposium on Network Cloud Computing and Applications (NCCA), pp.124–129, 2012.
[16] F. Bellard, “QEMU, a fast and portable dynamic translator,” USENIX Annual Technical Conference, 2005.
[17] H. Raj, K. Schwan, I. Ganev, and J. Xenidis, “High performance and scalable I/O virtualization via self-virtualized devices,” Proc. 16th IEEE International Symposium on High Performance Distributed Computing (HPDC ’07), pp.179–188, Monterey, CA, USA, 2007.
[18] M. Sridharan, A. Greenberg, N. Venkataramiah, Y. Wang, K. Duda, I. Ganga, G. Lin, M. Pearson, P. Thaler, and C. Tumuluri, “NVGRE: Network virtualization using generic routing encapsulation,” Internet draft, 2014.
[19] Offloading the Segmentation of Large TCP Packets, http://msdn.microsoft.com/en-us/library/windows/hardware/ff568840(v=vs.85).aspx
[20] L. Grossman, “Large receive offload implementation in neterion 10GbE Ethernet driver,” Proc. Linux Symposium, vol.1, pp.195–200, Ottawa, CA, July 2005.
[21] Red Hat Enterprise Linux 6 Virtualization Host Configuration and Guest Installation Guide, https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/pdf/Virtualization_Host_Configuration_and_Guest_Installation_Guide/Red_Hat_Enterprise_Linux-6-Virtualization_Host_Configuration_and_Guest_Installation_Guide-en-US.pdf
[22] Iperf, http://iperf.sourceforge.net/
[23] M. Casado, T. Koponen, S. Shenker, and A. Tootoonchian, “Fabric: A retrospective on evolving SDN,” Proc. First Workshop on Hot Topics in Software Defined Networking (HotSDN 2012), pp.85–90, Helsinki, Finland, Aug. 2012.

（平成 25 年 11 月 21 日受付，26 年 3 月 17 日再受付）

川島　龍太　（正員）

平成 19 岩手県立大・ソフトウェア情報学研究科博士前期課程了．平成 22 総合研究大学院大学・複合科学研究科了．博士（情報学）．同年株式会社 ACCESS 入社．平成 25 名工大大学院・工学研究科・助教，現在に至る．主に SDN，システムソフトウェアに関する研究に従事．情報処理学会，IEEE 各会員．

松尾　啓志　（正員）

昭和 58 名工大・情報卒．昭和 60 同大学大学院修士課程了．同年松下電器産業（株）入社．平 1 名工大大学院博士課程了．同年名工大・電気情報・助手．講師，助教授を経て，平 15 同大大学院教授，現在に至る．分散システム，画像認識，分散協調処理に関する研究に従事．工博．情報処理学会，人工知能学会，IEEE 各会員．