# 優先度を考慮した仮想スイッチにおける トンネル処理の負荷分散手法の検討

村松　真[†]　川島　龍太[†]　齋藤　彰一[†]　松尾　啓志[†]

† 名古屋工業大学大学院

**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，受信側の物理ホストにおいて特定の CPU コアにパケット処理が集中する場合がある．Receive Side Scaling(RSS) のようなマルチキュー NIC によりパケット処理を分散させる機能が普及しているが，割り込み先コアの衝突や VM との CPU リソース競合問題がある．そこで本研究では，物理 NIC からのハードウェア割り込み処理時に，ソフトウェア割り込み先のコアを適応的に決定して負荷を分散させる機能である VS-extend(Virtual Switch extend) を提案する．さらに，優先度の概念を導入し，特定のフローを優先的に低負荷な CPU コアで処理させる機能，及び VS-extend の設定をコントローラが一元的に行う仕組みも提案した．性能評価を行った結果，RSS を用いた場合はパケット処理の負荷が大きくなるにつれてスループットが低下したが，提案手法を用いた場合はスループットが向上した．また特定のフローに対して優先度を設定した場合，物理ホストの CPU がオーバコミットメント状態であっても優先フローの品質低下を防ぐことができた．

**キーワード**　Software-Defined Networking，ネットワーク仮想化，OpenFlow，仮想スイッチ，RSS

# Priority-based Load Distribution Method for Tunneling Processes in Virtual Switches

Shin MURAMATSU[†], Ryota KAWASHIMA[†], Shoichi SAITO[†], and Hiroshi MATSUO[†]

† Nagoya Institute of Technology, Graduate School of Engineering

**Abstract**　Edge-Overlay model constructing virtual networks using virtual switches and IP tunnel is promising in cloud datacenter networks. But software-implemented virtual switches can cause performance problems. Although multi queue functions like Receive Side Scaling(RSS) distrubute packet processing load onto multiple CPU cores, there are collision problems among IRQ cores and also PM-VM resources. In this paper, we propose VS-extend to address the problem. VS-extend adaptively determines soft-IRQ cores based on both the load and VM-pinning information. Futhermore, the proposed method supports priority-based CPU core selection, and the behavior of VS-extend can be managed by a unified controller. Performance evaluation results show that total throughputs of our approach was higher than a simple RSS-based model as packet processing load increases. In addition, we show that performance degradation of higher-priority flows can be prevented by proposed method.

**Key words**　Software-Defined Networking, Network Virtualization, OpenFlow, Virtual Switch, RSS

## 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では，マルチテナント方式を実現するために，各テナントは独立したネットワーク（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，テナントが持つ複数の仮想マシン（VM）と仮想ファイアウォールや仮想ルータなどの仮想的なネットワーク機器によって構成され，それぞれの機器は論理的なリンクによって接続されている．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フィックを論理的に分離する仕組みが必要となる．

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 NVO3 [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 トンネルを構築し，各仮想ネットワークのパケットに付加される固有の ID（トンネル ID）に

よってトラフィックを区別する．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る．加えて，仮想マシン上で仮想的なネットワーク機器をソフトウェアにより実現する Network Functions Virtualization(NFV) [4] の実用化が進められており，ますます仮想ネットワークの性能向上が求められている．

ところが，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（特にトンネリングや暗号化など負荷の大きい処理）が物理 CPU リソースを占有することで，VM が使用できるリソースが減少し，仮想ネットワーク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パケット処理を，物理 CPU リソースを考慮して分散させることで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となる．

これまで，ハードウェア支援によって仮想スイッチの処理負荷を低減させる手法が提案されてきた. [5] や [6] は，OpenFlow の処理を 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ている．また，Voravit ら [7] は，Intel®製の NIC に搭載されている FlowDirector [8] に仮想スイッチのフローエントリをキャッシングし，仮想スイッチにおける Look up 処理の負荷を低減させている．RSS [9] や Flow Director は，パケットヘッダの情報を用いてフローごとにハードウェア割り込み先の CPU コアを分散させ，特定のコアへの割り込みの集中を防ぐ．しかし，特定のハードウェア支援技術を用いることで，ベンダーロックインの問題が生じる場合がある．

そこで本研究では，特定の支援技術及びベンダ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である Virtual Switch(VS)-extend を提案する．提案システムでは，物理 NIC からのハードウェア割り込み (HW-IRQ) 処理時に OpenFlow 準拠のフローマッチング処理を行い，ソフトウェア割り込み (SW-IRQ) 処理を行う CPU コアを適応的に決定して負荷を分散させる．加えて，優先度の概念を導入し，特定のフローを優先的に低負荷な CPU コアで処理させる仕組み及び VS-extend の設定をコントローラが一元的に行うための制御プロトコルを設計する．

本論文では，受信側の物理ホストにおけるパケット処理の負荷分散方法と，物理 NIC のデバイスドライバ上で行った提案手法の実装について示す．加えて，OpenFlow コントローラと組み合わせた提案システム全体のアーキテクチャの設計についても示す．提案手法を用いて評価実験を行った結果，仮想スイッチにおけるパケット処理の負荷が高くなるにつれて，RSS を用いた場合はスループットが低下したが，提案手法を用いた場合はトンネル処理の負荷が分散され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，提案システムの全体的なアーキテクチャ及びドライバ内部での動作について，4. では，提案システムの内部実装について説明する．続いて 5. では，提案システムの性能評価実験について示す．最後に 6. 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

sNICh [5] や vswitch 処理の動的オフロード方式 [6] は，物理 NIC にフローテーブルを実装することで，仮想スイッチの OpenFlow 処理及びフレーム転送処理の負荷を低減させる．しかし，これらの技術は仮想スイッチをバイパスしてしまうため，高度な機能を仮想スイッチに集約するエッジ・オーバレイ方式への対応が困難である．

Voravit ら [7] は，Intel 製の NIC に搭載されている Flow Director [8] を用いて，仮想スイッチにおける Look up 処理を高速化し，物理ホストの CPU 負荷を低減させている．ただし，この技術はベンダ固有の技術であるため、多様なベンダの NIC が混在するデータセンタでは利用することが困難である．

Recive Side Scaling(RSS) [9] や Receive Packet Steering(RPS) [10] は，受信したパケットのヘッダ情報からハッシュ値を計算し，その値を基にプロトコル処理を行うコアを決定し，パケット処理の負荷を分散させる．しかし，ハッシュ値は機械的に算出されるため、割り込み先コアの衝突が発生し，特定のコアに高負荷なフローが集中する可能性がある．それに加え，VM の使用コアが予め指定されているピニング [11] で負荷の大きいパケット処理を行うと VM の CPU リソースが奪われ，仮想ネットワーク性能を低下させてしまうという問題も挙げられる．

## 3. 提案手法のアーキテクチャ

本章では， 2. で述べた課題を解決するため，仮想スイッチでのパケット処理の負荷分散手法と OpenFlow コントローラから一元的に管理するアーキテクチャについて説明する．

図 1 に提案手法の全体アーキテクチャを示す．図中の VS(Virtual Switch)-extend は，物理 NIC のデバイスドライバ内に実装され，Core Table と Flow Table を参照し，フローの SW-IRQ 先コアの決定及び SW-IRQ 制御を行う．また，コントローラは各物理ホスト及び VM 情報をもとに，OpenFlow の拡張メッセージを用いて VS-extend の FlowTable へ負荷分散を行うためのエントリを追加する．

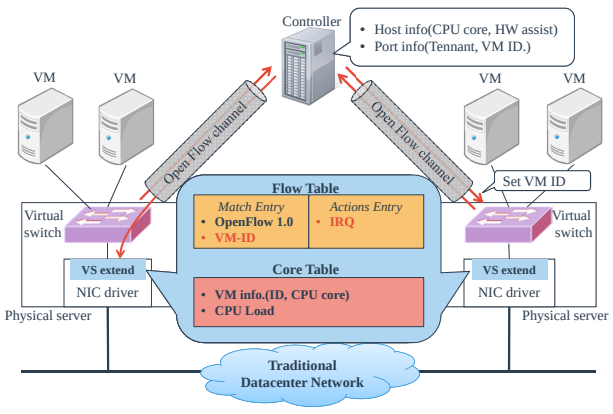


図 1　提案手法の全体アーキテクチャ

Fig. 1　Architechtual overview of the proposed system

### 3.1 Linux におけるパケット受信処理

提案手法の負荷分散手法を説明する前に，Linux カーネルにおけるパケット受信処理について説明する．

通常，Linux のネットワークドライバは，HW-IRQ が行われた後，同一 CPU コア上でポーリング処理を行い，物理 NIC 内のキューに残されたパケットを取り出す．その後，Linux カーネルでは HW-IRQ が行われたコアに対して SW-IRQ を行い，TCP/IP や VXLAN などのプロトコル処理を行う．仮想スイッチが存在する場合は，SW-IRQ の後，必要ならば OpenFlow などのパケット処理を行い，パケットを宛先へ転送する．しかし，Linux カーネルは HW-IRQ が行われた，または RPS により機械的に決定されたコアに対して SW-IRQ を行うため，特定のコアに負荷が集中する場合がある．そこで本論文では，フローごとに SW-IRQ を行う CPU コアを適切に選択し，パケット処理の負荷を分散する手法を提案する．

### 3.2 Flow Table/Core Table の詳細

図 1 に示した VS-extend は，受信フローを識別して割り込み先コアを決定するための Flow Table と，物理ホストのコア情報を管理する Core Table を内部に持つ．

Flow Table は OpenFlow1.0 に基づいたエントリマッチに加え，VXLAN などのトンネリングプロトコルでカプセル化されたパケットヘッダをマッチエントリとして扱うことが可能である．さらに，アクション処理に IRQ を追加することにより，マッチしたフローは IRQ で指定されたコアに SW-IRQ を行うことが可能となる．条件にマッチしない場合は，従来通り HW-IRQ が行われたコアで SW-IRQ を行う．

Core Table は各コアにピニングされている VM の識別子（VM ID）と負荷情報を管理しており，条件にマッチするフローが到達した時，VS-extend は Core Table を参照しフローの SW-IRQ 先コアを決定する．これにより，物理ホストの負荷，VM のピニング状態に応じてパケット処理を分散できる．

### 3.3 コントローラへの物理ホスト/VM 情報の登録方法

本節では，提案手法における仮想スイッチと OpenFlow コントローラ間のプロトコルについて説明する（図 2）．提案手法では，仮想スイッチとコントローラが OpenFlow コネクションを確立した後，仮想スイッチはベンダ拡張メッセージである `OFPT_VENDOR` メッセージを送信し，仮想スイッチが動作している物理ホストのコア数，RSS や FlowDirector などの機能の有無を伝達する．次に，VM が起動すると，`OFPT_PORT_MOD`

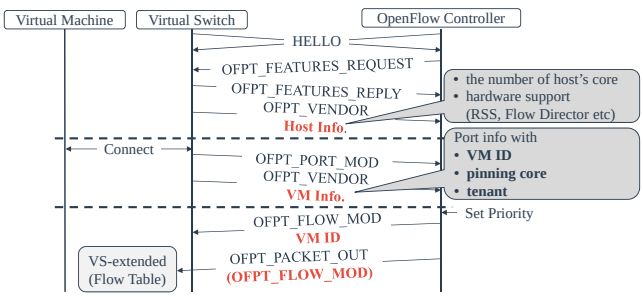


図 2　ホスト情報の登録及び VS-extend 設定の流れ

Fig. 2　A protocol between a vswitch and an OpenFlow controller

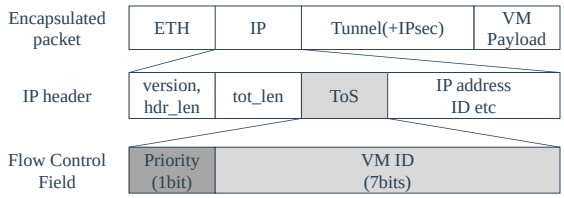


図 3　フロー制御ヘッダの構造

Fig. 3　A flow control feild structure

メッセージが送られるが，提案手法ではそれに加え，VM が属するテナント，VM ID，VM のピニングコアも送信する．これらの情報をもとに，コントローラは提案手法の使用の是非を判断し，VS-extend 内の Flow Table の設定を行う．さらに，`OFPT_PACKET_OUT` メッセージによって `OFPT_FLOW_MOD` メッセージをカプセル化することでコントローラから VS-extend の Flow Table へエントリを追加する．

### 3.4 フロー制御ヘッダ

暗号化された及びフラグメントされたフローは，既存の OpenFlow マッチングでは識別が不可能である．そこで提案手法では，特定のフローを識別するために，IP ヘッダの ToS フィールドのセマンティクスを変更し，Flow Control Field(FCF) を定義した．図 3 に示すように，特定 VM 宛のフローを識別する VM ID(7bit) と優先フローを識別する Priority(1bit) が存在する．IP トンネリングを用いる場合，通常は内部 IP ヘッダの ToS フィールドの情報をそのまま外部 IP ヘッダに引き継ぐため，トンネリング機能と IPsec を併用しても，外部 IP ヘッダの ToS フィールドからフローを識別することができる．また，フラグメントしたパケットに対しても FCF は引き継がれるため，VM ID を参照することでフラグメント化されたパケットを全て同一コアで処理させることも可能である．VM ID が 0 の場合は，宛先 VM は未指定であり，他のフィールドを用いてマッチング処理を行う必要がある．Priority フラグがセットされいる場合，優先的に VM がピニングされていないかつ，負荷の低い CPU コアを選択することで，優先フローのパフォーマンスの低下を抑えることが可能となる．

### 3.5 提案手法のパケット受信処理

以上の節で述べた提案手法によるパケット受信処理の流れを以下にまとめる．

（1） 物理 NIC がパケットを受信した時，特定の CPU で HW-IRQ を行い，ネットワークドライバがパケットを受け取る．

（2） パケットヘッダと VS-extend の Flow Table でフローマッチング処理を行う．

（3） マッチングに成功した場合，アクション処理の `IRQ` に従い，そのコアへ SW-IRQ を行う．もし割り込み先が決定されていない場合は，Core Table を参照して割り込み先コアを決定する．

（4） マッチングに失敗した場合，HW-IRQ が行われたコアに割り込む．

### 3.6 VS-extend における追加ルール

コントローラが設定するフローエントリの追加ルールを表 1 に示す．仮想スイッチに対しては，従来の OpenFlow によるフ

表 1　追加ルール
Table 1　Additional rules

| | |
|---|---|
| vSwitch | `Actions:set_flow_control_field` |
| VS-extend | `Match:VM_ID, Actions:IRQ` |

ロー設定の他，FCF を設定するアクション処理も必要に応じて行う．VS-extend の Flow Table に対しては，負荷分散を行うフローに対するマッチングルールを設定する．

## 4. 実　　装

提案手法である VS-extend の機能を MellanoxConnectX(R)-3 のドライバ上に実装した．本章では，フローの SW-IRQ 先を制御する方法に焦点を当てて，その内部実装について説明する．

図 4 に提案手法で実装する Mellanox ドライバ内のフローチャートを示す．RPS 機能が有効である場合，`netif_receive_skb` 関数でカーネルにパケットが渡された後，`rps_get_cpu` 関数によって SW-IRQ 先コアを指定し，`enqueue_to_backlog` 関数で指定したコアに対して SW-IRQ 処理を行う．なお `rps_get_cpu` 関数は，受信パケット中の IP アドレス及びポート番号などから算出される skb 構造体の rxhash 値を用いて割り込み先コアを機械的に決定する．

一方，提案手法では，パケットをカーネルへ渡す前に，ドライバ内でフローの割り込み先を制御する．はじめに `match_entry` 関数で VS-extend の Flow Table とマッチング処理を行う．マッチしたエントリ内で SW-IRQ 先コアが指定されていない場合，`set_flow_entry` 関数により，Core Table を参照して SW-IRQ 先コアを決定し，エントリに追加する．例えば，Core Table が表 2 のエントリをもつ場合，フローの SW-IRQ 先は，VM がピニングされていないかつ負荷が閾値（ここでは 50 とする）以下の `core2` に決定され，Flow Table に Match とともに Actions に `IRQ:2,rxhash:skb->rxhash` が設定される．デフォルトでは，HW-IRQ が行われたコアの負荷が高くても SW-IRQ 先は変更されないが，提案手法では SW-IRQ 先を負荷の小さいコアへ変更することが可能となる．ただし，全てのコアの負荷が閾値より高い場合は，HW-IRQ を行ったコアを使用する．次に，`set_sock_table` 関数で，ハッシュ値と CPU コア番号の関係を保持する `sock_flow_table` に `skb->rxhash` とコア番号を格納する．この `sock_flow_table` はカーネルから提供されており，`get_rps_cpu` 関数から `skb->rxhash` を用いて参照することで SW-IRQ 先を指定する．以降のパケットは `set_skb_hash` 関数

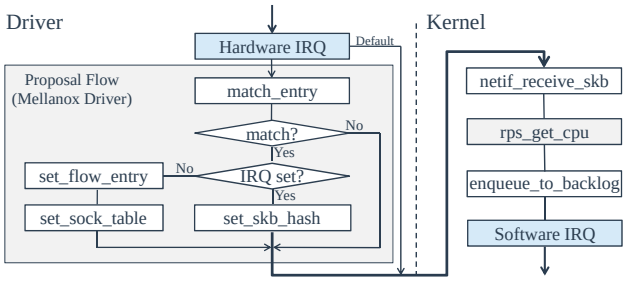


図 4　Mellanox ドライバ内のフレーム処理フロー
Fig. 4　A flowchart of frame processing in Mellanox Driver

表 2　Core Table と Flow Table のステータス
Table 2　Statuses of Core Table and Flow Table

Core Table

| core | status |
|---|---|
| 0 | `load:10, VM pinning, VM_ID:1` |
| 1 | `load:50, VM pinning, VM_ID:2` |
| 2 | `load:20` |
| 3 | `load:50` |

Flow Table

| Match | Actions |
|---|---|
| `nw_src=10.0.0.1,VM_ID=1` | `IRQ:3,rxhash:45678` |
| `nw_src=10.0.0.1,VM_ID=2` | `IRQ:-,rxhash:-` |

によって，エントリで指定されたハッシュ値に `skb->rxhash` 値を書き換えることで同一コアへ SW-IRQ を行う．上記の実装はすべて Ethernet ドライバ内で行うため，既存の Linux カーネル実装を変更する必要はない．

## 5. 性能評価

本章では，実装した提案手法を利用した際の VM 間通信の性能を評価する．はじめに予備評価として，負荷の大きいパケット処理（ここでは VXLAN と AES(注1) を使用）を行う場合に，RSS において割り込み先コアの衝突が発生した場合と，VM とフロー処理（FP）のリソースが競合する場合のスループットを測定した．次に，2 組の VM 間通信を対象に，提案手法による負荷分散の効果を評価する．最後に，3 組の VM 間通信を対象とし，一組の VM 間通信優先度の高いフローと設定し，優先フローのパフォーマンス低下の抑制を評価する．

図 5 と表 3 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 [12] を用いた．また，パケットサイズが 64，1400，8192bytes の場合についてそれぞれ評価した．

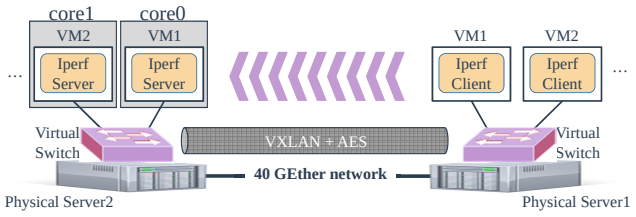


図 5　評価環境
Fig. 5　Experimental environment

表 3　マシン仕様
Table 3　Machine specifications

| | Physical Server1 | Physical Server2 | VM |
|---|---|---|---|
| OS | CentOS 6.5(2.6.32) | | Ubuntu 12.04(3.5.0) |
| CPU | Core i5(4 core) | Core i7(4 core) | 1 core(Pinning) |
| Memory | 16 GB | | 2GB |
| Buffer | 4 MB | | |
| Network | 40Gbit Ethernet | | - |

(注1)：本評価では，VXLAN overIPsec を使用する場合を想定する

表 4　予備評価結果 [Mbps]
Table 4　Preliminary evaluation[Mbps]

| packet size | 64bytes | 1400bytes | 8192bytes |
|---|---|---|---|
| flow-collision | 39.5 | 954.9 | - |
| non-flow-collision | 39.4 | 1509.9 | 1702.4 |
| FP-VM collision | 19.7 | 617.3 | 4.0 |
| non FP-VM collision | 19.8 | 754.6 | 880.1 |

### 5.1　予 備 評 価

実験では，Iperf クライアントが Iperf サーバに対して 80 秒間連続的に UDP パケットを送信し，Iperf サーバの中 60 秒間のスループットの平均を合計したものを評価値とする．2 組の VM 間通信を対象にして，RSS において割り込み先コアの衝突が発生するケース，1 組の VM 通信を対象にして，VM がピニングされているコアでフロー処理を行うケースについてそれぞれ測定した結果を表 4 に示す．

1400bytes の場合，割り込み先コアの衝突が発生しないケースのスループットは 1509.9Mbps であるが，衝突時ではスループットが 954.9Mbps と低下している．また，8192bytes でも，衝突が発生しないケースでは 1702.4Mbps と 2 つの VM 間通信を確立できているが，衝突発生時では通信が確立できない結果となった．同様に FP と VM の CPU リソース（FP-VM）が競合する場合，1400bytes，8192bytes の時，仮想スイッチによるフロー処理のために VM のリソースが奪われてしまうため，スループットが低下している．以上の結果から，負荷の大きいパケット処理において，コアの衝突，FP-VM の競合がスループット低下につながることがわかった．

### 5.2　2 対 2 通信における性能評価

次に，2 対 2 通信における受信側であるサーバのスループットの合計を時間経過と共に評価した．実験では，Iperf クライアントが 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フローを 20 回生成する．Iperf クライアントは，フローを生成するごとにランダムにソースポートを決定するため，引数でソースポートを指定できるように変更した．また，ポート番号は予め乱数で生成したデータを用意し，全てのモデルに対して同じファイルを用いて測定を行った．RSS を用いる場合は，フローごとに割り込み先コアが変化するため，受信側物理

表 5　各コアにおけるソフトウェア割り込み回数
Table 5　The total amount of SW-IRQ per CPU core

| **64bytes** | core0 | core1 | core2 | core3 |
|---|---|---|---|---|
| default | 0 | 0 | 0 | 165,159,804 |
| rss | 39,936,135 | 53,710,644 | 41,317,702 | 28,924,990 |
| proposal | 0 | 0 | 82,634,610 | 82,634,450 |
| **1400bytes** | | | | |
| default | 0 | 0 | 0 | 109,134,696 |
| rss | 35,263,529 | 48,655,872 | 41,435,459 | 29,002,358 |
| proosal | 0 | 0 | 82,873,766 | 82,875,742 |
| **8192bytes** | | | | |
| default | 0 | 0 | 0 | 184,126,030 |
| rss | 197,691,348 | 5,391,984 | 6,221,397 | 10,844,689 |
| proposal | 0 | 0 | 171,580,236 | 171,738,991 |

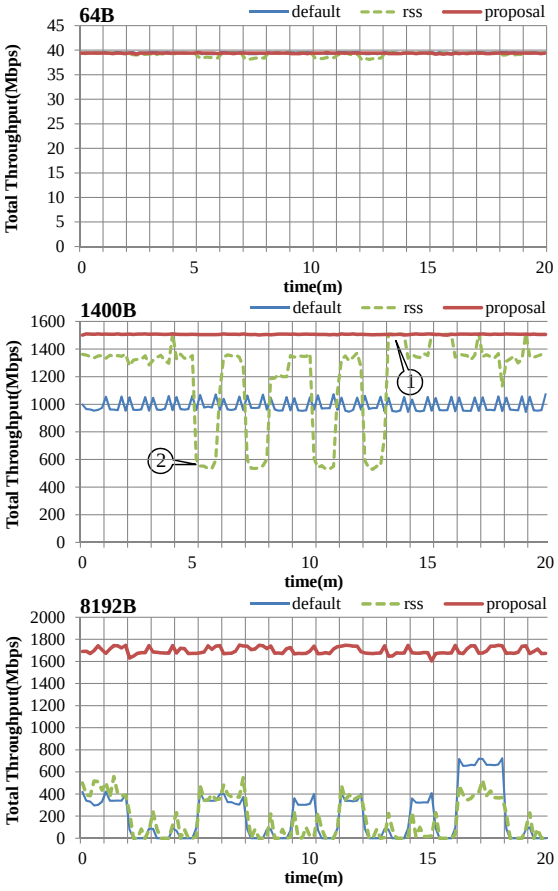


図 6　2 対 2 通信におけるトータルスループット
Fig. 6　Total Throughput of two VMs communication

サーバではフロー処理，FP-VM の競合が発生する可能性がある．提案手法を用いる場合は，NIC ドライバ（Mellanox 2.2.0）に搭載されている RSS を併用する，また，ドライバ内のフローテーブルは予め設定されている．(VM1 のフロー：core2，VM2 のフロー：core3)

図 6 に，2 対 2 通信における使用帯域幅を集約した結果を示す．図中の”default”は RSS/RPS を使用しない場合，”rss”は RSS 機能を有効にした場合，”proposal”は提案手法（+RSS）を表している．64bytes の場合は，それぞれのモデルで大きな差はないが，1400bytes では，default のスループットが 1000Mbps，rss のスループットはフローごとに変動していることがわかる．図中の 1 では，rss により 2 つのフローが別々のコアに割り込み負荷を分散したためスループットが向上しているが，2 では VM2 がピニングされているコアで競合が発生したため，default の性能を下回っている．さらに，8192Bytes の場合，default，rss のスループットは大きく低下しているが，proposal では 1700Mbps のスループットを維持していることがわかる．

次に，各コアにおけるソフトウェア割り込み回数を回数を表 5 に示す．RSS では VM がピニングしているコアにも SW-IRQ を行なっており，また 8192bytes の場合は，core0 への割り込みが極端に集中している．これはフラグメント化されたパケットは IP ヘッダのみでハッシュ値を割り出すため，core0 への割り込み回数が増大したと考えられる．一方，提案手法では，core2,core3 へのみ SW-IRQ が行われるため，VM とフロー処

表 6　各 VM におけるパケット欠落率 [%]
Table 6　Packet loss of VMs[%]

| | 64bytes | | 1400bytes | | 8192bytes | |
|---|---|---|---|---|---|---|
| | VM1 | VM2 | VM1 | VM2 | VM1 | VM2 |
| default | 0.1 | 0.1 | 34.5 | 34.5 | 51.4 | 52.4 |
| rss | 0.8 | 0.9 | 15.8 | 17.0 | 95.3 | 43.5 |
| proposal | 0.1 | 0.1 | 0.2 | 0.2 | 14.8 | 14.7 |

理のリソースが競合せず，スループットの低下を抑えている．

最後に，パケットサイズごとに各モデルのパケット欠落率を表 6 に示す．表 6 の 1400，8192bytes をみると，提案手法のパケット欠落率が大きく削減できていることがわかる．一方，特に 8192bytes における default と rss では，通信が確立できていない結果となった．これは特定のコアに割り込みが集中し，物理ホストでのパケット処理が追いつかずパケットが破棄されてしまうことが原因だと考えられる．

### 5.3　優先度設定による優先テナントの性能評価

最後に 3 対 3 通信において優先度を設定した場合の評価を行った．比較対象として RSS を用い，提案手法を用いる場合は，優先フローを扱う VM1 のフロー処理は VM がピニングされていないコア core3 で行い，それ以外の VM のフロー処理は自 VM がピニングされているコアで行う．表 7 に各 VM の 20 分間におけるスループットの平均を示す．

1400bytes の結果を見ると，RSS では各 VM のスループットは平均して 450Mbps 程度であるが，提案手法では，VM1 のスループットが 745Mbps と他の VM と比べ性能が向上していることがわかる．さらに，8192bytes の場合，RSS は全ての VM の性能が低下しているが，提案手法では，VM1 のスループットが 844Mbps と全く性能が低下していないことがわかった．

### 5.4　考　　察

前節までの結果から，提案手法は，負荷の大きいパケット処理を適切なコアに分散させることで性能を向上できることがわかった．そのため，提案手法はエッジ・オーバレイ方式など，高度なパケット処理を仮想スイッチへ移行するネットワークにおいて特に有効であると言える．一方，RSS や RPS などの仕組みを用いてパケット処理の負荷を分散させることは可能であるが，特定の状況下で性能が大幅に低下することがある．しかし，提案手法を用いることで物理ホストがオーバコミットメント状態であっても，コントローラから優先フローを設定することで優先テナントの品質の低下を抑制できることがわかった．これは，課金体系のデータセンタにおいて，コントローラから優良なテナントに対して多くの物理リソースを割り当てる場合に有効である．

表 7　優先度を考慮した性能評価 [Mbps]
Table 7　Throughput of considering the priority

| | 64bytes | | | 1400bytes | | |
|---|---|---|---|---|---|---|
| | VM1 | VM2 | VM3 | VM1 | VM2 | VM3 |
| rss | 19.4 | 19.5 | 19.3 | 481.8 | 465.6 | 406.3 |
| proposal | 19.7 | 19.7 | 19.7 | 745.3 | 635.0 | 646.4 |

8192bytes

| | VM1 | VM2 | VM3 |
|---|---|---|---|
| rss | 3.4 | 192.8 | 20.9 |
| proposal | 844.8 | 466.3 | 181.4 |

## 6. む　す　び

本論文では，特定のハードウェア支援技術に依存せずに，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分散する VS-extend を提案した．VS-extend は，ネットワークドライバ上で，OpenFlow の Match 処理を参考にフローの識別を行い，フローごとに仮想スイッチの処理を行うコアを指定しパケット処理の負荷を分散させる．加えて，コントローラから一元的に VS-extend の設定を行うアーキテクチャも設計した．実際に性能評価を行った結果，提案手法ではフロー処理を適切に分散させ，優先度を設定した場合，優先フローのスループットが低下せず，最大のネットワークパフォーマンスを発揮することが確認できた．

今後の課題として，設計したコントローラと仮想スイッチ間のプロトコルの実装と合わせて，優先テナントのリソースを動的に確保する手法及び VS-extend の実装をドライバ外で行う手法の確立が挙げられる．

**謝辞**　本研究の一部は，科研費基盤研究 (C)24500113 による．

### 文　　献

[1] D. Black, J. Hudson, L. Kreeger, M. Lasserre, and T. Narten, “An architecture for overlay networks(NVO3),” Aug. Internet Draft. 2013.
[2] 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] SDN and OpenFlow World Congress, `http://portal.etsi.org/nfv/nfv_white_paper.pdf`, “Network functions virtualization,” Oct. 2012.
[5] K.K. Ram, J. Mudigonda, A.L. Cox, S. Rixner, P. Ranganathan, and J.R. Santos, “sNICh: Efficient last hop networking in the data center,” Proc. ACM/IEEE Symposium on Architectures for Networking and Communications Systems (SNCS 2010), pp.1–12, Oct. 2010.
[6] 辻　聡，飯星貴裕，狩野秀一，“vswitch 処理の動的オフロード方式の実装と評価,” 電子情報通信学会技術研究報告. NS, ネットワークシステム，vol.111，no.43，pp.69–74，2011.
[7] V. Tanyingyong, M. Hidell, and P. Sjodin, “Using hardware classification to improve pc-based openflow switching,” Proc. IEEE 12th International Conference on High Performance Switching and Routing (HPSR 2011), pp.215–221, July 2011.
[8] “Flowdirector”. `www.kernel.org/doc/Documentation/networking/ixgbe.txt`.
[9] “Receive-side scaling enhancements in windows server 2008,” Nov. 2008. Internet Draft.
[10] “Receive packet steering”. `https://www.kernel.org/doc/Documentation/networking/scaling.txt`.
[11] L. Abeni, C. Kiraly, N. Li, and A. Bianco, “Tuning kvm to enhance virtual routing performance,” Proc. IEEE International Conference on Communications (ICC 2013), pp.3803–3808, June 2013.
[12] “Iperf”. `http://iperf.sourceforge.net/`.