Vol.2016-OS-138 No.20
2016/8/9

SDNスイッチにおける柔軟なパケット制御機構の提案

小嶋 奨[,a)]　福田 浩章[,b)]

概要：Software Defined Network（SDN）はネットワークをソフトウェアで定義することによって制御する技術である．SDN はパケットの転送機能を有する複数のスイッチと，パケットのルーティングを管理するソフトウェアを物理的に分離する．そして，両者を共通プロトコルで接続することで，ネットワークの機能をコントローラーで一元管理し，プログラムで柔軟に変更することができるため注目されている．

SDN はスイッチとそれを制御するコントローラで構成される．コントーラがスイッチを制御するための標準プロトコルである OpenFlow では，スイッチはパケットが保持する情報のうち OSI 基本参照モデルのレイヤ 4 以下の情報に従った機能しか利用することができない．しかし，ネットワーク機能にはレイヤ 5 以上の情報を用いる次世代ファイアウォールや，L7 ロードバランサの持つ機能が存在する．これらのレイヤ 5 以上の情報に従った機能を利用する方法は，パケットをコントローラに送ることや，専用機能を有するネットワーク機器を配置することで実現できる．しかし，前者ではコントローラに頻繁にパケットを送る必要があるため通信速度の低下が発生し，後者では SDN の利点であるネットワーク構築の柔軟性を欠くこととなってしまう．

そこで，本稿では，コントローラを介さずにスイッチでパケットのレイヤ 5 以上の情報に従った動作を行うための機構を提案する．本機構により，コントローラとの通信で発生する速度の低下を低減することが期待できる．

キーワード：Software Defined Network，OpenFlow，Open vSwitch

How to Prepare Your Paper for IPSJ Journal (version 2016/07/13)

Susumu Kojima[,a)]　Hiroaki Fukuda[,b)]

Abstract: A benefit of Software Defined Networks is flexible and faster management of network flows made possible by the execution of flow-specific actions based upon inspection of various packet fields. However, current switches limit the inspected fields to layer 2-4 headers. This cannot flow-handling that uses higher-layer information without sending the packets to the controller. Sending the packets to controller produces switch-to-controller round-trip delays. This paper proposes actions that enables packet-handling that uses L5-L7 information on switch. We implement Check-action that is prototype-action using L5-L7 information on Open vSwitch. The results show that Check-action has low overhead. Moreover, we consider the architecture enable to be specified based upon inspection of L5-L7 information in other ways.

Keywords: Software Defined Network，OpenFlow，Open vSwitch

1. はじめに

SDN[1], [2]/OpenFlow[3] は，パケット転送機能を有する複数のスイッチと，ソフトウェアでパケットのルーティングを管理する単一のコントローラで構成される．SDN は，

†1 現在，芝浦工業大学
Presently with Shibaura Institute of Technology
a) ma16044@shibaura-it.ac.jp
b) hiroaki@shibaura-it.ac.jp

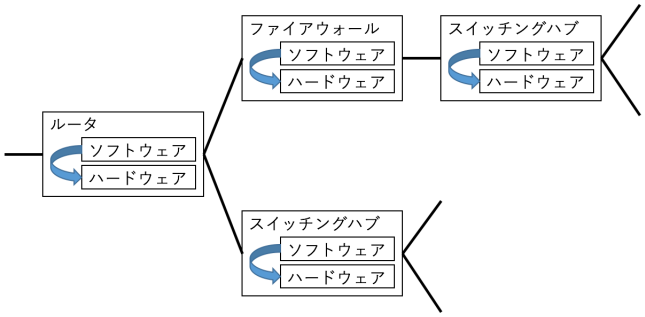

図 1 従来のネットワークの構成例

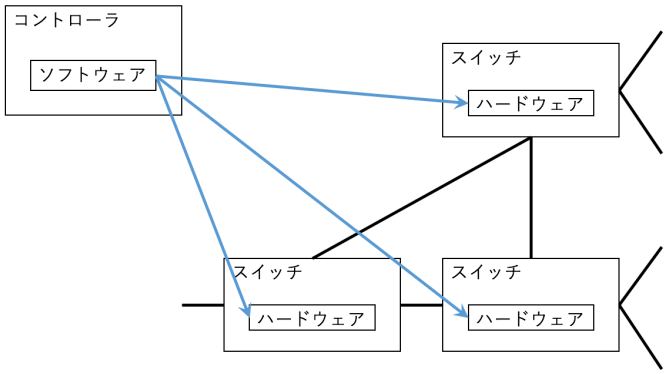

図 2 SDN の構成例

プログラム可能なソフトウェアにより，コントローラからから共通のプロトコル（OpenFlow）を用いてスイッチを一元管理することで，従来の機器に必要な個別の設定やトポロジ設計に捕らわれることなく，柔軟で迅速なネットワークの構築を可能にする．具体的には，SDN では，パケットの条件と，条件にマッチした時の振舞いをコントローラに予め記述し，未知なパケットがスイッチに到達した時にコントローラに問い合わせることでコントローラからスイッチにルーティング情報が送信され，パケットが転送される．そして，ルーティング情報はそのままスイッチに記憶されるため，同一の条件を満たすパケットはスイッチに記憶された情報を利用して転送されるため，コントローラに問い合わせることはない．しかし，現在の SDN ではスイッチが参照できるのはパケットのレイヤ 4 までの情報であるため，レイヤ 5 以上の情報を必要とする次世代ファイアウォールや，L7 ロードバランサをスイッチだけで実現することはできない．そのため，それらを実現するにはその都度コントローラに問い合わせて一度限りのルーティング情報をスイッチに転送する，もしくは専用機器を導入せざるを得ない．前者の方法は，その都度コントローラにパケットを転送する必要があるためオーバヘッドがかかる．後者は SDN 非対応の機器が導入されることによって，ソフトウェアでネットワークを管理できるという SDN の設計思想と反する．

そこで本研究では，コントローラを経由することなくスイッチでレイヤ 5 以上の情報に従ったルーティングを行う機構を提案する．本研究では，スイッチでパケットの条件にマッチした時実行される処理（アクション）に着目し，レイヤ 5 以上の情報を参照する Check アクションを導入する．そして，Check アクション導入による影響を，通常のパケット転送時と，Check アクション導入時のスループットとレイテンシを測定し，評価する．また，本機構を拡張し，検査対象のパケットをより細かく指定できるためのアーキテクチャを考察する．

2. SDN/OpenFlow

2.1 SDN

SDN は，ソフトウェア定義でネットワークを制御する技術を指し，ネットワーク仮想化を実現する技術の一つである．SDN はネットワークの制御機能を担うコントロールプレーンと，データ転送機能を担うデータプレーンの 2 つで構成される．一般に，コントロールプレーンにあたるハードウェアをコントローラ，データプレーンにあたるハードウェアをスイッチと呼ぶ．

従来のネットワークでは図 1 のように，ルータやスイッチングハブ，ファイアウォールなど，それぞれ個別のネットワーク機能を持った機器を組み合わせ，運用目的に合わせて適切に配置する必要がある．図 1 の構成例では，パケットはまずルータを通り，ファイアウォールを経由するルートと経由しないルートを通り，それぞれスイッチングハブを通してその先へと送られる．従来の機器は個別のネットワーク機能を実現するため，図 1 の各機器のように，それぞれのハードウェアとそれを制御するソフトウェアを個別に備えていた．従来の機器のハードウェアおよびソフトウェアはそれぞれの機器で異なり，各機器の設定に関しても機器毎に個別の管理ツールを用いて設定を行う必要があった．そのため，ネットワーク全体のルーティングポリシーやトポロジを変更するにはそれぞれの機器を設定し直す必要があり，多くの時間や人出を費やす必要がある．

一方，SDN では，図 2 のように複数のスイッチと一つのコントローラから構成される．スイッチはメッシュ状に配置され，コントローラは全てのスイッチと通信を行う．SDN に対応したスイッチはパケット転送を行うハードウェアだけを有し，未知のパケットに対するルーティングの制御はコントローラに委譲する．そして，この制御を行うためのプロトコルを定義している．その結果，コントローラで多数の SDN スイッチを一元管理できるため，目的に応じた柔軟な制御の提供と迅速な切り替えが可能になる．そのため，SDN では，ネットワーク構成を変更するために必要

Match Field	Priority	Counters	Instructions	Timeouts	Cookie
dst_ip = 192.168.0.1, src_ip = 192.168.0.11	priority = 3	n_packet = 3	actions = output : 1	hard_timeout = 300	cookie = 0x0000000000000001
dst_ip = 192.168.0.2, src_ip = 192.168.0.12	priority = 1	n_packet = 0	actions = drop	idle_timeout = 150	cookie = 0x0000000000000001
src_ip = 192.168.0.13	priority = 1	n_packet = 0	actions = all	idle_timeout = 100, hard_timeout = 300	cookie = 0x0000000000000002
⋮	⋮	⋮	⋮	⋮	⋮

表 1 フローテーブル

な時間や人手が少ない．例えば，図 2 の構成例では，左下のスイッチにルーティング機能とファイウォール機能を割り振り，右の 2 つのスイッチにスイッチング機能を割り振るといったネットワークや，左下のスイッチにはルーティング機能を割り振り，右上のスイッチにファイアウォール機能とスイッチング機能を，右下のスイッチにスイッチング機能を割り振るといったネットワークを構築でき，前者のネットワークから後者のネットワークへの構成の切り替えも即座に行うことができる．

2.2 OpenFlow

2.1 節で述べたように，SDN ではスイッチとコントローラで通信し，パケット処理の委譲やルーティングテーブルの書き込みを行うため，標準プロトコルが必要となる．そのために現在一般的に利用されているのが OpenFlow である．OpenFlow では，後述するフローエントリの書き込みによってコントローラがスイッチを制御する．フローエントリは一定の条件にマッチするパケットに対する動作を示した規則であり．スイッチはこのフローエントリに従ってパケットを処理する．スイッチはフローエントリを登録するフローテーブルを保持し，コントローラはスイッチのフローテーブルにフローエントリを書き込むことでスイッチを制御する．

2.3 フローエントリ

フローテーブルは図 1 のように表現することができ，複数のフローエントリを保持している．フローエントリは図 1 に示すように，Match Field，Priority，Counters，Instructions，Timeouts，Cookie の 6 つの要素で構成されている．この要素のうち，Match Field は，物理ポート番号，MAC アドレス，IP アドレス，ポート番号などの条件を複数設定することができる．Match Field に設定された条件を満たすパケットは Instruction に従って処理される．

Instruction ではパケットの書き換えや任意の物理ポートからのパケットの転送などの複数のアクションを設定することができる．

例えば，図 1 の一番上のフローエントリは，送信元 IP が 192.168.0.11 で送信先 IP が 192.168.0.1 のパケットに対して物理ポートから転送を行うという規則を示している．

上記の通り，Match Field はフローエントリがどのようなパケットに対して実行される規則であるかを示すが，指定できるのは OSI 基本参照モデルにおけるレイヤ 1 からレイヤ 4 に当たる情報についての条件のみである．そのため，HTTP プロトコルヘッダやデータ部分に含まれる情報などのレイヤ 5 以上の情報を必要とする一部のファイアウォール機能や負荷分散をフローエントリで表現することができず，スイッチでこれらの処理ができない．現在，スイッチで処理のできない機能を SDN で実現する方法は次の 2 つがある．

(1) コントローラに処理を移譲
(2) 専用機能を備えた従来の機器を導入

コントローラに処理を移譲する場合，コントローラとスイッチの間でパケットの送受信を行う必要があり，ネットワークの遅延に繋がる．更に，通信が頻繁に発生し，コントローラが行う処理が増加するため負荷が大きくなる．また，SDN ではネットワークの制御部分と，ネットワーク機能を実行する部分を分離した形態をとっているため，ネットワーク機能の処理を制御部分であるコントローラで行うことは避けるべきである．

専用機能を備えた従来の機器を導入する場合，従来の機器は SDN コントローラで制御することができず，ネットワークの構成に制限が発生する．また，設定も個別に行う必要がある．そのため，ネットワーク構成の変更を迅速かつ柔軟に行うことができるという SDN の利点を損なうことになる．

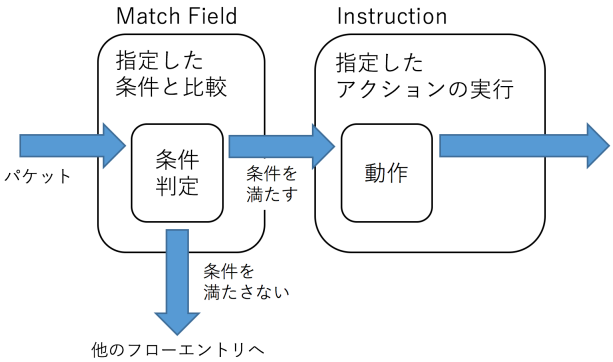

図 3 Match Field で条件を判定するときのパケットの処理手順

3. 提案

本稿では，スイッチでレイヤ 5 以上の情報を参照してルーティングを行う機構を提案する．スイッチでレイヤ 5 以上の情報を必要とする機能を実現する方法は 2 つ考えられる．

(1) Match Field を拡張
(2) アクションの処理中に条件を判定

それぞれの方法について以下で説明する．

3.1 Match Field を拡張

Match Field を拡張する場合，スイッチでのパケットの処理手順は図 3 に示すように，Match Field に設定した条件との比較を行い，一致すれば Instruction で指定したア

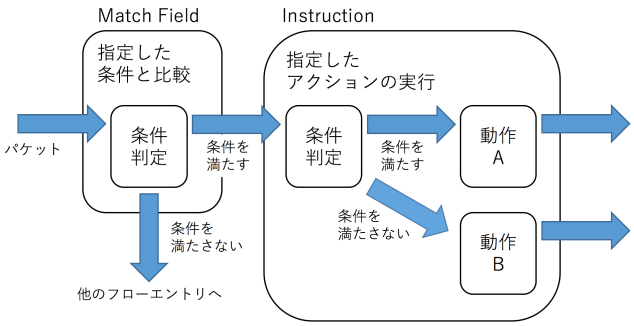

図 4 アクションで条件を判定するときのパケットの処理手順

クションの動作を実行する．この方法は，Match Field に設定する条件を加えるだけであり，条件の判定は今までと同じく Match Field で行われるため自然である．しかし，Match Field はスイッチに到達した全てのパケットに対して探索を行うため，探索アルゴリズムを用いて探索回数を減らす必要がある．

3.2 アクションの処理中に条件を判定

アクションの処理中に条件を判定する場合のスイッチでのパケットの処理手順を図 4 に示す．まず，Match Field での条件判定が行われ，そこで条件を満たした一部のパケットのみがアクションの処理中にレイヤ 5 以上の情報の条件判定を行う．そして，その結果に応じた動作を実行する．この方法は，既に Match Field での条件判定を通過した一部のパケットのみがレイヤ 5 以上の情報を参照するため，処理速度への影響も少ないと考えられる．しかし，Match Field で行うべき条件判定を Instruction で行うことになるため，フローエントリの各要素の機能分担には反している．

そこで，本研究では，アクションの処理中に条件判定を行うことで，スイッチでレイヤ 5 以上の情報を参照してルーティングを行う機構を実現する．

4. 実装

本研究では，後述する Check アクションを OpenFlow に対応したオープンソースの仮想スイッチである Open vSwitch-2.3.2[5] に実装した．Open vSwitch と，実装した Check アクションについて以下で述べる．

4.1 Open vSwitch

Open vSwitch は OpenFlow をサポートするオープンソースのソフトウェアスイッチである．XenServer, KVM, VirtualBox などの仮想化プラットフォームでもよく利用されている．OpenFlow 以外にも NetFlow，sFlow，VLAN などにも対応している．Linux 標準のブリッジと互換性があり，代替ブリッジとして機能させることができる．また，スイッチの監視や管理を行うためのコマンドも備えている．Open vSwitch は大半がプラットフォーム独立である

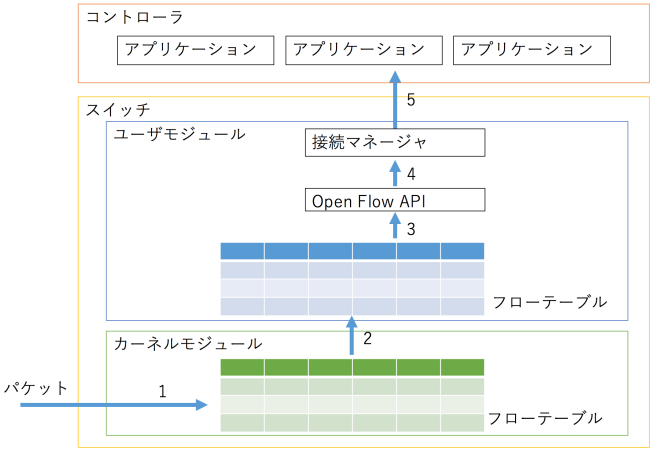

図 5 Open vSwitch のアーキテクチャ

C 言語で記述されているため，移植性が高い．本稿の機構はこの Open vSwitch に実装する．

4.2 Open vSwitch のアーキテクチャ

Open vSwitch のアーキテクチャを図 5 に示す．図 5 に示すように Open vSwitch はカーネルモジュールとユーザモジュールで構成され，どちらもフローテーブルを保持している．まず，図 5 の（1）のように，パケットがスイッチに到達するとカーネルモジュールのフローテーブルで参照され，パケットと一致する Match Field を持つフローエントリが存在する場合は，カーネルモジュールで処理が行われる．カーネルモジュールのフローテーブルには，ユーザモジュールで以前に一致したパケットの完全一致な Match Field 情報を持つフローエントリがキャッシュされている．

カーネルモジュールのフローテーブルに，一致する Match Field が存在しなかったパケットは，図 5 の（2）のように，ユーザモジュールのフローテーブルで参照が行われる．ユーザモジュールでも，同様にフローテーブルの参照を行い，一致した場合はカーネルモジュールへフローエントリのキャッシュと Instruction の処理を行い，一致しなかった場合はコントローラへの問い合わせが行われる．

コントローラへの問い合わせは，図 5 の（3），（4）のように，OpenFlow API を利用して接続マネージャを呼び出し，接続マネージャがパケットををカプセル化して，図 5 の（5）で示すようにコントローラへ送信する．

コントローラはソフトウェアに従ってパケットを検査し，必要なスイッチに対してフローエントリの書き込みやパケットの送信を行う．

Open vSwitch でのパケットの処理はこのようにカーネルモジュール，ユーザモジュール，コントローラの順で行われ，処理にかかる時間もこの順で速い．特に，コントローラでの処理はパケットの送信やカプセル化など，より多くの時間を必要とする．

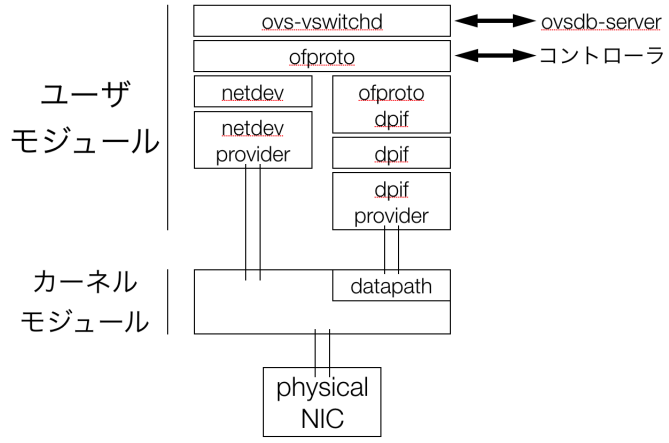

図 6 Open vSwitch のモジュール構造

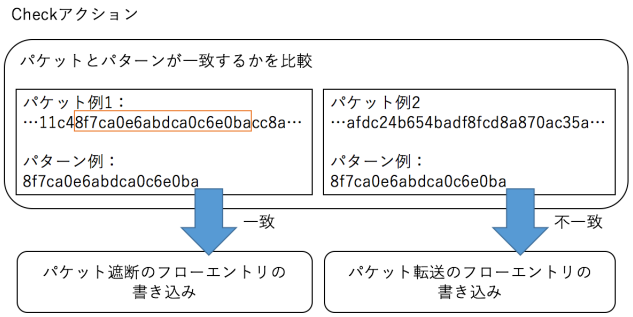

図 7 Check アクションの動作

4.3 Open vSwitch のモジュール構造

Open vSwitch のモジュール構造を図 6 に示す．datapath では図 5 のカーネルモジュールに相当する動作が実装されている．カーネルモジュールでのキャッシュフローテーブルの動作などの変更を行う場合はこのコンポーネントに実装する．

ofproto dpif では，図 5 のユーザモジュールでのフローテーブルやその探索，アクションの実装が行われている．

ofproto に図 5 の OpenFlow API や接続マネージャが実装されている．このコンポーネントによって Open vSwitch はコントローラとの通信を行っている．

本研究では，アクションの追加を行うため，アクションの実装が行われている ofproto dpif に Check アクションの実装を行う．

4.4 Check アクション

Check アクションの処理の流れは図 4 の指定したアクションの実行と同様に行われる．図 7 に Check アクションのパケット検査における動作を示す．Check アクションはパケットのデータを引数として受け取り，パケットのデータからレイヤ 5 以上の情報も参照して，バイト列のパターンとの比較による条件判定を行う．そして，この結果に応じて条件分岐を行い，受け取ったパケットとバイトコードのパターンが一致していた場合には指定した動作を行うフローエントリを書き込む．一致しなかった場合にはパケットを遮断を動作として指定したフローエントリを書き込む．その後，もう一度パケットをフローテーブルのフローエントリと比較し直すことで，書き込んだフローエントリの処理を適用させる．パターンとして既知の不正パケットのパターンを登録することで，既知の不正パケットを遮断するセキュリティアクションとして利用することができる．

フローエントリの書き込みの処理，書き込むフローエントリの動作の指定方法は Nicira Extension の learn アクション（フローエントリの書き込みを行うアクション）と同様に行う．また，フローテーブルのフローエントリとパケットを比較し直すアクションは OpenFlow1.3 以降で使用可能な go_to_table アクション，もしくは，Nicira extension[4] の resubmit_table（パケットを再度フローテーブルと比較するアクション）アクションを用いる．Nicira Extension は旧 Nicira（現 VMware）による OpenFlow の拡張である．

CPU	メモリ
Intel Core i5（4 コア，2.60GHz）	8GB（1600MHz DDR3）

表 2 ホストマシン

CPU	メモリ	アダプタータイプ
4 コア	1GB	Intel PRO/1000 T Server（82543GC）

表 3 仮想マシン（クライアント，サーバ，スイッチ）

5. 実験と評価

Check アクションでは，forwarding に比べ，パケットのデータとバイト列のパターンの比較をするため，オーバヘッドがかかる．なので，スイッチの処理性能にどれだけの影響があるかを調べるため，スループットとレイテンシを計測する実験を行った．

5.1 実験環境

実験はクライアント，サーバ，スイッチの 3 つの仮想マシンを Oracle VirtualBox[6] で起動し，1Gbps で接続して計測を行った．ホストマシンの設定を 2，仮想マシン（クライアント，サーバ，スイッチ）の設定を 3 に示す．

5.2 スループット

Check アクションを設定したスイッチと，forwarding を設定したスイッチに対して，それぞれ TCP パケットをクライアントからスイッチを経由してサーバへ送信するスループットを計測した．計測には，Tcpreplay[7] を使用し，12,000 パケット（19,860,000 bytes）のパケットをそれぞれ，接続帯域 1Mbps から 100Mbps の範囲で指定して送信した．

forwarding については，1Mbps から 100Mbps の全ての帯域でほぼ 100 %のスループットを示していることがわ

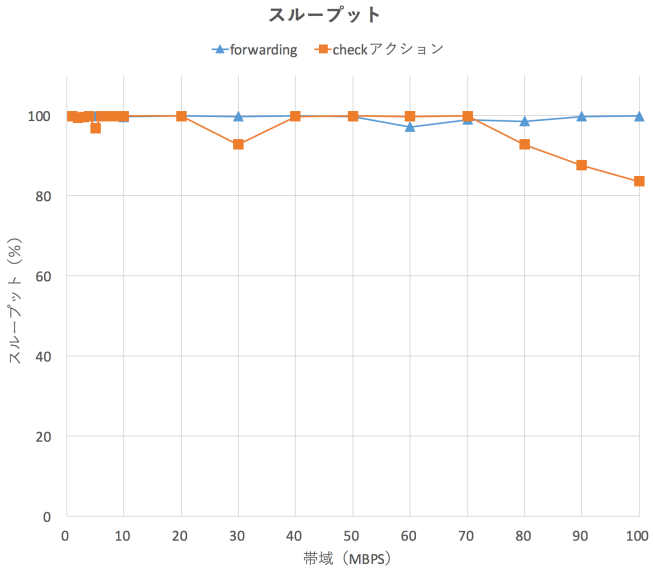

図 8 スループット

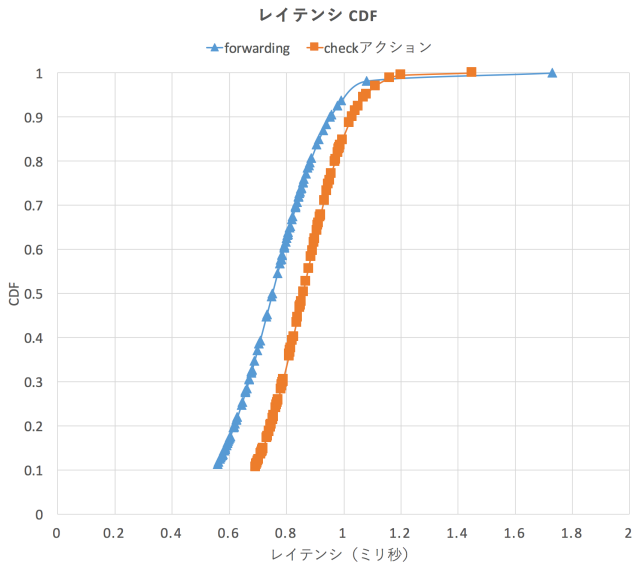

図 9 レイテンシ

かる．

一方，Check アクションは，forwarding と比べると帯域 80Mbps 以降のスループットが下降していることがわかる．また，帯域 30Mbps でも差が発生しているが，帯域 40Mbps から帯域 70Mbps までは通常の転送と遜色ないスループットを示している．

このことから，スループットに関して，70Mbps までは Check アクションの処理速度はパケットの転送以上と同等の速度を出しているが，70Mbps を超えたあたりから Check アクションの処理がパケットの転送に追いつかなくなり，帯域を使い切るだけの速度でパケットを処理できていないことがわかる．

5.3 レイテンシ

レイテンシの計測は ping メッセージを，Check アクションを設定したスイッチと，forwarding を設定したスイッチに対して，クライアントからスイッチを経由してサーバへ送信し，ラウンドトリップタイムを計測して行った．ping メッセージは 100 回送信を行い，ラウンドトリップタイムの最小値と最大値を除いたものを用いてレイテンシを計算した．

縦軸の CDF は累積分布関数を示し，ping メッセージのラウンドトリップタイムが横軸で示す時間以内である割合を示している．

forwarding では 6 割のパケットが 0.8 ミリ秒以内のラウンドトリップタイムである．また，8 割のパケットはラウンドトリップタイムが 0.9 秒以内であることがわかる．

一方，Check アクションのラウンドトリップタイムを通常の転送と比べると，6 割のパケットが 0.9 ミリ秒以内，8 割のパケットが 1.0 ミリ秒以内であり，1 割近くラウンドトリップタイムが大きくなっている．

このことから，Check アクションは通常の転送と比べて，一回あたりの ping メッセージを処理するのに 1 割ほどの遅れが発生していることがわかる．

5.4 評価

スループットに関して，今回の計測では，100Mbps でも 80 %以上のスループットを出すことができている．また，レイテンシに関しては 1 割ほどの遅れとなっている．一般的なネットワークにおいて，100Mbps での通信を行う際，上限である 100Mbps で通信することは多くないと考えられる．また，Web サーバへのアクセスなどでは，長時間の間ネットワークの帯域を上限まで使いきり続ける状況は想定しにくい．よって，提案機構の処理性能への影響は少ないと言える．

提案機構は計測により処理性能への影響が少ないことがわかったが，3 に記述したようにフローエントリの各要素の機能分担には反している．よって，今後の課題として Match Field の拡張についても検討する必要がある．

5.5 Match Field の拡張

Match Field では，スイッチに到達したパケット全てに対して探索が行われる．線形探索では，フローエントリの数に比例して膨大な処理時間が必要となるため，OpenFlow 対応スイッチでは，探索アルゴリズムを用いることで探索回数を減らしている．

提案機構の実装を行った Open vSwitch では探索アルゴリズムにハッシュ法を用いている．しかし，Match Field の拡張を行う場合，ハッシュ法では処理できない条件も存在する．Check アクションで行っているパケットのデータとバイト列のパターンの比較もその一つである．

そこで，Match Field をマルチレイヤ構造化する方法を考えた．マルチレイヤ構造では，ハッシュ法に比べると速度は低下するが，同じフローエントリの登録数でもメモリの使用量が少ないというメリットも存在する．メモリの容量が少ない機器を OpenFlow に対応させる場合に利用することや，削減したメモリを別の処理に使うことが考えられる．

6. 関連研究

Application-aware Data Plane Processing in SDN[8] では，コントローラの機能をスイッチでも実行可能にし，コントローラの機能を実行するアプリケーションをスイッチにインストールすることで，コントローラと通信することなくスイッチでコントローラと同等の制御を可能にしている．レイヤ 5 以上の情報をスイッチで処理するには有効な方法であるが，ネットワークの制御部分とデータ転送部分を分けた SDN の構造に反している．提案機構は，コントローラの機能をスイッチに組み込むのではなく，従来ではスイッチでは行うことのできないデータ転送の機能のみを提供するため，SDN の構造には反していない．

OFX[9] では，セキュリティアクションを適用する場合，しばしばレイヤ 5 以上に相当する情報を必要とする状況があることに触れ，スイッチを経由することで性能の低下を引き起こすことを防ぐために，セキュリティモジュールをスイッチに追加インストールするためのフレームワークを提案している．OFX は，スイッチで単独で動作することができず，OpenFlow コントローラを必要とする．提案機構は，スイッチ単独でも動作することができ，OFX に比べてより多くの場面で利用が可能である．

UNISAFE[10] では，ユーザモジュールだけでなくカーネルモジュールへの機能の追加や強力なサービスチェーン，複数のセキュリティアクションの実装を行っている．レイヤ 5 以上の情報をスイッチでとても高速に処理することができるが，Instruction の処理で条件を判定している．また，アクションの変更はプログラムを書き換える必要がある．UNISAFE は，処理性能が高いが，3.2 節で述べたフローエントリの機能分担について考慮されていない．

7. まとめ

本稿では，SDN スイッチでのより柔軟なパケット制御を実現するためにレイヤ 5 以上の情報を参照する Check アクションを提案した．Check アクションの処理性能は，forwarding と比べて，レイテンシは 1 割ほどの遅れ，100Mbps のネットワークでのスループットは約 84 %であった．

今後の課題としては，まず，Match Field のマルチレイヤ構造化の実装を行い，その有用性について実験を行う．Match Field をマルチレイヤ構造化することで Match Field が拡張され，条件を増やしても速度の低下を抑えることができると考えられる．

また，メモリの容量や CPU の処理性能など，環境によるレイヤ 5 以上の情報を参照するアクションの有用性の変化や，Match Field のマルチレイヤ構造化との性能の差異を実験で評価し，アクションとして実装を行うべき条件の洗い出しを行う必要もある．

参考文献

[1] OPEN NETWORKING FOUNDATION ”Software-DefinedNetworking: The New Norm for Networks White Paper” April 13, 2012.
[2] OPEN NETWORKING FOUNDATION
https://wwww.opennetworking.org/
[3] OpenFlow
https://www.opennetworking.org/sdn-resources/openflow
[4] Nicira extension
http://git.openvswitch.org/cgi-bin/gitweb.cgi?p=openvswitch;a=blob;f=include/openflow/nicira-ext.h
[5] Open vSwitch
http://openvswitch.org/
[6] Oracle VM ViatualBox
https://www.virtualbox.org
[7] Tcpreplay
http://tcpreplay.appneta.com
[8] H. Mekky, F. Hao, S. Mukherjee, Z. L. Zhang, and T. V. Lakshman
Application-aware Data Plane Processing in SDN. In *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, 2014.
[9] J. Sonchack, A. J. Aviv, E. Keller, and J. M. Smith
Poster: OFX: Enabling openflow extensions for switch-level security applications. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, 2015.
[10] T. Park, Y. Kim, and S. Shin
UNISAFE: A Union of Security アクション s for Software Switches. In *Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, 2016.

正誤表

論文の内容に誤りがありましたので，下記の通り訂正いたします.

訂正箇所	1 ページ目中央下，英文タイトル
誤	How to Prepare Your Paper for IPSJ Journal
正	Flexible packet-handling on SDN switch