Vol.2017-IOT-39 No.11
Vol.2017-SPT-25 No.11
2017/9/29

分散型SDN制御プレーンへのDoS攻撃の緩和手法の提案

松尾 亮輔[1]　今泉 貴史[2]

概要：SDN では，大規模なネットワークを制御するために，複数のコントローラによる分散制御を行う手法がいくつか提案されている．また，ネットワーク内のトラフィックの偏りに対応するために，スイッチの管理権限をコントローラ間で移動させる手法が提案されている．しかし，この手法は少数のスイッチの集団にトラフィック量が集中する場合，効果的に機能しないことがある．また，近年では SDN-DoS 攻撃という，帯域を圧迫せず効果的に制御プレーンのリソースを消費させる手法も提案されている．複数のコントローラによる分散型制御プレーンにおいて，この攻撃を緩和するために，コントローラ間で処理を分担する手法を提案し，性能の検討を行う．

キーワード：SDN, DoS, コントローラ

A Proposal for Relaxing Denial of Service Attacks on Distributed SDN Control Plane

Ryosuke Matsuo[1]　Takashi Imaizumi[2]

1. はじめに

ネットワークを管理する技術の 1 つに，SDN (Software Defined Network) がある．SDN は，ネットワークを構成しデータ転送を行うデータプレーンと，制御命令をデータプレーンへ適用するコントロールプレーン，経路制御などの各種判断を行うアプリケーションによって構成される．コントロールプレーンは，コントローラと呼ばれる機器によって構成される．管理者はネットワークの制御ルーチンをアプリケーションを通して定義することで，ネットワーク全体を一元的に管理することができる．この管理上の利点から，SDN は論理ネットワークの構築などに利用されている．

しかし，ネットワーク全体の制御をコントローラを介して行うため，コントローラには大きな負荷がかかり，これを意図的に利用した SDN-DoS 攻撃が発生することが懸念されている．通常制御時の負荷への対策として，複数のコントローラによるネットワークの分散制御手法が提案されているが，ほとんどがネットワークのトポロジーに基づいた処理範囲の分割を行っている．そのため，攻撃者がトラフィックを局所的に集中させた場合，その処理による負荷は一部のコントローラに集中してしまう．

本研究では，攻撃発生時に攻撃パケットの処理を，稼働している他のコントローラへ分散させるシステムを提案する．複数のコントローラを最大限活用することで，SDN-DoS 攻撃により発生する大量のパケット処理へのスループットを向上させることを目的とする．

2. OpenFlow

SDN における，制御プレーンとデータプレーン間の通信の仕様を定めた代表的なプロトコルに，OpenFlow がある．OpenFlow では，データプレーンを構成する OpenFlow スイッチは，フローテーブルと呼ばれる命令表を持つ．フローテーブルの要素であるフローエントリには，対象とするパケットの条件と，パケットの処理方法の組が格納さ

[1] 千葉大学大学院融合科学研究科
Graduate School of Advanced Integration Science, Chiba University
[2] 千葉大学統合情報センター
Institute of Management and Information Technologies, Chiba University

れている．OpenFlow スイッチはパケットを受信すると，ヘッダを解析し，フローテーブル内に条件を満たすフローエントリが存在するか検索する．条件を満たすフローエントリが存在しなかった場合，処理方法がわからないため，テーブルミスとなる．

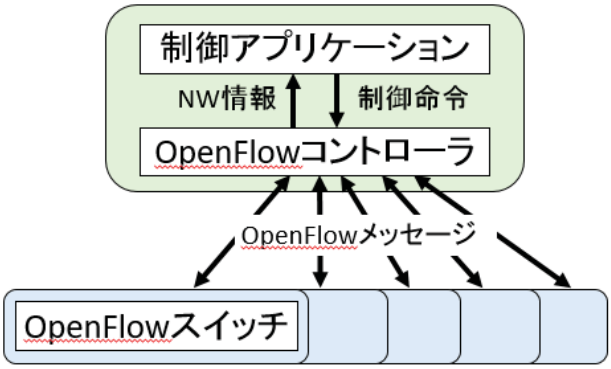

図 1　OpenFlow の一般的な構成

テーブルミスが発生した際のデフォルトの動作は，OpenFlow のバージョンによって異なる．OpenFlow1.2 までは，テーブルミスを起こしたパケットを基に，Packet_In という問い合わせメッセージを生成し，コントローラに送信した．これに対し，OpenFlow1.3 以降では，テーブルミスを起こしたパケットは破棄される．ただし，リアクティブ型と呼ばれる制御方式では，コントローラが受け取った Packet_In に対応するフローエントリを生成し，元のスイッチに対して適用することで，動的にネットワークの制御を行う．そのため，リアクティブ型で制御を行う場合，管理者は特定のパケットの問い合わせをコントローラへ送信させるように設定する必要がある．

3. 分散型制御プレーン

OpenFlow はコントローラ—スイッチ間の仕様であるため，コントローラの構成については決められていない．そのため，NOX [1] や Trema [2] など多くのコントローラフレームワークでは，1 台のコントローラインスタンスによってネットワーク全体の制御を行う設計となっている．しかし，数百台のサーバーが相互に接続するデータセンターや，物理的に広範囲に及ぶ WAN のような，規模の大きいネットワークの管理をする際には，制御プレーンの物理的な集約は性能や遅延の観点から難しい．

制御プレーンのスケーラビリティを向上させるために，複数のコントローラによる分散型制御プレーンを構成する手法がいくつか提案されている．[3], [4]

これらの手法では，データプレーンを複数のエリアに分割し，1 つのエリアを 1 台のコントローラが管理するという構成をとる．コントローラは自身が管理するエリア内のトポロジー情報を収集し，フローを制御する．自身のエリア外の情報（宛先ホストの位置など）が必要な場合，他のコントローラやデータベースへ問合せることで，必要な情報を得る．1 台のコントローラが管理するスイッチの数を少なくすることで，コントローラの管理する規模を，性能に応じた適切な範囲に収めることが出来る．

分散型制御プレーンの構成は，並列型と階層型に大きく分けられる．[5] 並列型では，コントローラのインスタンス同士が，協調してネットワークの制御を行う．並列型のコントローラ構成の代表的なものとして，Onix や ONOS [6] が挙げられる．階層型では，スイッチと接続する各コントローラが，コントローラを統括する上位のコントローラへ接続する構成をとる．

多くの分散型制御プレーンの提案では，それぞれのコントローラが管理するエリアは静的であり，稼働中にそのエリアは変化しない．しかし，実際のネットワークを流れるトラフィックの量は，時間や空間的な要因によって動的に変化する．[7] そのため，一部のエリアへトラフィックが集中した場合，そのエリアを管理するコントローラに過負荷がかかることが考えられる．

4. SDN-DoS

近年，SDN の構成要素を標的とした攻撃に対するセキュリティが注目されている．[8] その中で，制御プレーンに対する攻撃手法の 1 つとして，SDN-DoS と呼ばれる手法が指摘されている．攻撃者は，対象となる OpenFlow スイッチのフローテーブルの内容を推測し，コントローラへの Packet_In メッセージの送信を誘発させるようなパケットを大量に送信する．このパケットはコントローラへの大量の問合せを発生させ，受信したコントローラのリソースの枯渇や，応答の遅延，パケットロスを引き起こす．コントローラが Packet_In に対して応答できない場合，ネットワーク内のフローは正しい宛先へルーティングされず，制御が失われる．

フローエントリはパケットのヘッダを対象パケットを指定するための条件に用いるため，攻撃パケットはヘッダの値を調節した小さいサイズのパケットとなる．そのため，攻撃者が十分な量の攻撃パケットの生成能力を持つ場合，OpenFlow スイッチのパケット転送能力が攻撃の規模の上限となる．商用の OpenFlow スイッチでは，100M パケット/s の送信能力を持つものがあり，中規模以上のデータセンター向けの製品では，1000M パケット/s の送信能力を持つスイッチもある．[9] 既存の研究では，OpenFlow コントローラの Packet_In メッセージの処理能力は，シングルコアで動作するフレームワークである NOX では 30K/s [10]，マルチスレッドによるパフォーマンスの向上を図った手法である Maestro でも，600K/s となっている．[11] これらの値は，Packet_In を処理する制御アプリケーションの実装や，コントローラを稼働させるサーバーマシンの性能に

よって変化するが，SDN-DoS による大量の攻撃パケットの処理は，1 台のコントローラの処理能力を容易に超えるものだと言える．

この攻撃が，分散型の制御プレーンによって管理される SDN に対して行われた場合について考える．攻撃パケットがネットワーク全体で均一に発生する場合，攻撃によって発生する処理量は，すべてのコントローラによって均等に分散される．しかし，実際には攻撃者は特定のホストから攻撃を行うため，そのホストが接続しているスイッチに攻撃パケットが集中することになる．また，外部からの攻撃の場合も，ネットワークの境界となるスイッチに攻撃パケットが到着することになる．そのため，攻撃によって引き起こされる大量の Packet_In メッセージは，特定のコントローラに集中して送信される可能性がある．この場合，攻撃パケットの処理は一部のコントローラだけが行うことになり，SDN-DoS 攻撃への耐久力は少数のコントローラの処理能力に依存してしまう．

5. 関連研究

コントローラの負荷のバランスを取るために，各コントローラが管理するエリアを動的に変更する手法が提案されている．[12,13] これらの手法では，ネットワーク稼働時の負荷に基づいて動的にコントローラが管理するエリアを変更する．また，各コントローラの負荷を監視し，必要に応じて新たなコントローラの追加や，余分なコントローラの削除も行う．これらの動作を通じて，1 台のコントローラにかかる負荷を，十分な処理能力を確保出来る量に抑え，各コントローラの負荷を均一にすることを目的とする．

ElastiCon のような，負荷によってコントローラの管理エリアを変更する手法を用いる場合，攻撃を受けているコントローラが管理するスイッチの数を減らすことで，負荷の軽減が出来る．しかし，管理エリアの調整はスイッチ単位のため，攻撃パケットの到着が少数のスイッチに集中している場合，対象となるスイッチを管理するコントローラに，依然として過負荷が発生してしまう．

OpenFlow の仕様では，1 台のスイッチへ複数のコントローラが接続出来るが，Packet_In メッセージは，スイッチへのフルアクセス権を持つ全てのコントローラへ転送される．そのため，攻撃パケットは接続する全てのコントローラへ送信されてしまう．処理を行う Packet_In メッセージをコントローラ間で相互に分担することが出来れば，1 台のスイッチによる負荷の分散が可能となるが，そのためには処理を行うメッセージの調整アルゴリズムや，同期のためのメカニズムが必要となる．これらの処理はコントローラへ追加の負荷を与えることになるため，効果的ではない．

6. 提案手法

本研究では複数のコントローラによる分散制御下において，SDN-DoS 攻撃によるパケットの処理を全てのコントローラへ分散させることで，攻撃の影響を緩和するシステムを提案する．このシステムでは，スイッチとコントローラに加えてオーケストレータという機器を追加する．オーケストレータはすべてのコントローラと接続を確立する．(図 2)

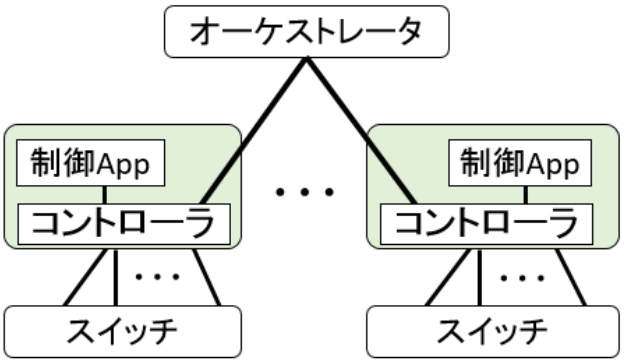

図 2 提案手法の構成

オーケストレータは，ネットワーク全体の情報収集と，各コントローラの CPU 使用率，Packet_In メッセージの到着頻度の監視を行う．ネットワーク情報は各コントローラから収集し，コントローラは制御の過程で必要に応じて情報の要求が出来る．また，各コントローラの CPU 使用率や Packet_In レートはコントローラ毎の負荷の指標として利用する．CPU の使用率が閾値を超えた場合，SDN-DoS 攻撃を受けていると判断し，負荷分散モードへの移行を対象のコントローラへ通知する．負荷分散モードでは，攻撃の対象となっているコントローラが受信する Packet_In メッセージを，オーケストレータを経由して稼働中の他のコントローラへ転送する．

このシステムでは，複数台のコントローラに処理を分担させ，攻撃対象のコントローラは Packet_In の転送と，処理結果をスイッチへ反映させるという単純な動作のみ行うことで，より多くのパケットを処理できるようにする．また，メッセージの分配やネットワーク情報の同期などの作業は専用のオーケストレータが担当し，負荷分散によって発生する処理を，コントローラの負担を軽減する．これらの処理の分担によって，SDN-DoS 攻撃による大量のメッセージの処理に対してより多くの計算リソースを確保し，ネットワーク全体の制御スループットを向上させることを目標とする．

6.1 システムの動作

本提案手法では，稼働しているコントローラの負荷に基づき，通常の分散制御と負荷分散モードのどちらかでネットワーク制御を行う．(図 3) この節では，通常制御時と負荷分散モード時，およびそれぞれのモードへの移行時のシ

ステムについて，オーケストレータと攻撃対象のコントローラ，それ以外のコントローラが行う動作を解説する．

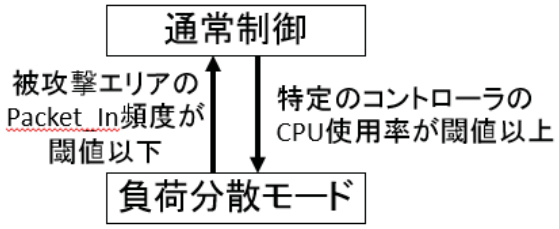

図 3　提案手法の状態遷移

6.1.1　通常制御

このシステムでは通常時，並列型の分散制御プレーンによるネットワークの制御を行う．各コントローラは，自身が管理するエリア内のスイッチと接続し，リンク，ホスト等のトポロジー情報を収集する．収集したトポロジー情報は，自身のデータベースへ保存すると同時に，オーケストレータへ転送される．そのため，オーケストレータはネットワーク全体の正確なトポロジー情報を持つ．コントローラが Packet_In メッセージの処理を行う過程で，他のコントローラが制御するエリアの情報が必要な場合，従来の並列型制御のように他のコントローラへ情報の要求メッセージを送信することもできるが，この手法ではオーケストレータに情報の要求を行うことも出来る．

また，オーケストレータは定期的に全てのコントローラへ CPU 使用率を確認するメッセージを送信し，コントローラの負荷状況を確認する．オーケストレータは同時に，コントローラ毎の Packet_In メッセージの到着頻度も確認する．この値も負荷の指標として利用するが，Packet_In の頻度とコントローラの負荷の相関は，コントローラ内の制御アプリケーションの実装などによって変化するため，負荷分散モードへの移行判断には用いない．特定のコントローラの CPU 使用率が閾値を超えた場合，オーケストレータは Packet_In の集中による過負荷が発生したと判断し，負荷分散モードへの移行を開始する．オーケストレータはこの時点での Packet_In の頻度の値を保存し，通常制御へ戻る際の閾値の設定に利用する．

6.1.2　負荷分散モードへの移行

負荷分散モードへの移行は，オーケストレータからコントローラへの通知メッセージの送信によって開始される．通知の種類は，攻撃の対象となっているコントローラと，それ以外のコントローラで区別される．攻撃対象ではないコントローラには，通知メッセージの送信と同時に，攻撃が発生しているエリアのトポロジー情報を送信する．この動作によって，全てのコントローラは攻撃対象エリアの Packet_In を処理するために必要な情報を，自身のメモリに格納する．攻撃の対象となっているコントローラは，通知を受信すると，それ以降スイッチから受信する Packet_In メッセージを，オーケストレータへ転送するように動作を変更する．この一連の動作を終えると，システムは負荷分散モードによる制御へ移行する．

6.1.3　負荷分散モード

負荷分散モードでは，攻撃対象コントローラへ届く Packet_In はすべてオーケストレータへ転送される．オーケストレータは受信した Packet_In を，他のコントローラへ順番に分配する．各コントローラは，Packet_In メッセージの内容と攻撃対象エリアのトポロジー情報に基づいて処理を行い，Flow_Mod メッセージ (フローエントリの変更命令) などの出力が存在する場合，そのメッセージをオーケストレータへ返送する．オーケストレータは受信した出力を攻撃対象コントローラへ転送する．攻撃対象コントローラは受信した出力を解析し，本来の宛先となるスイッチへメッセージを送信する．

Packet_In メッセージの中には，エリア内のリンク検出のための LLDP パケットや，ホスト検出のための ARP パケットが含まれることがある．これらのメッセージによって攻撃対象エリアのネットワーク情報が更新される場合，通常制御時と同様に更新情報をオーケストレータへ転送する．Packet_In メッセージを正確に処理するためには，最新のトポロジー情報が求められるため，オーケストレータは定期的に各コントローラへトポロジー情報の変化を通知し，更新を行わせる．

オーケストレータは負荷分散モード時も，各コントローラの CPU 使用率と Packet_In の頻度の監視を行う．攻撃対象以外のコントローラの CPU 使用率が，処理に影響を及ぼす恐れのある値になった場合，そのコントローラは転送先から除外し，CPU 使用率の値を観察する．攻撃対象コントローラの Packet_In メッセージの受信頻度が閾値以下まで下がるのを確認すると，過負荷が発生する状態は解消されたと判断し，通常の制御への移行を開始する．

6.1.4　通常制御への移行

オーケストレータが通常制御への移行を決定すると，始めに攻撃対象コントローラへ通知が送信される．攻撃対象コントローラは通知を受信すると，Packet_In の転送を止め，オーケストレータへ転送終了メッセージを送信する．以降は，攻撃対象コントローラが直接 Packet_In の処理を行う．オーケストレータは転送終了メッセージを受け取り，自身の Packet_In の転送キューが空になったことを確認すると，各コントローラへ通常モードへの移行を通知する．各コントローラは攻撃対象エリアの Packet_In 要求が残っていないことを確認し，移行完了メッセージを返送する．オーケストレータがすべてのコントローラから移行完了メッセージを受け取った時点で，通常モードへの移行が完了したと判断し，各コントローラへの攻撃対象エリアのトポロジー更新情報の通知を終了する．

コントローラには攻撃対象エリアのトポロジー情報が

保存されたままだが，以降更新は行われず，そのエリアのPacket_In メッセージも届かないため，タイムアウトなどによって消滅する．ただし，制御アプリケーションの実装によっては，制御の過程で攻撃対象エリアの古いホスト情報を参照してしまう可能性もある．このような場合のために，オーケストレータは通常制御への移行時に，ホスト情報やリンク情報などを更新し，明示的に消滅させることも出来る．

7. 考察

前章では，従来の並列型分散制御プレーンの構成に新たにオーケストレータを追加し，負荷分散を行うシステムを提案した．この章では負荷分散モードと従来の制御を比較し，提案手法が性能に対して与える影響について考察する．

遅延：負荷分散モードでは，攻撃対象コントローラが受信した Packet_In メッセージを，オーケストレータを介して他コントローラへ転送し，処理を行わせる．そのため，従来の制御と比較して，追加で 4 ホップ分の転送が発生する．これにより，提案手法ではスループットの向上とのトレードオフとして，追加のレイテンシが加わる．また，管理するネットワークが地理的に広範囲に分散している場合，コントローラ間の距離による遅延も発生する．そのため，地理的に近いコントローラの集団毎に 1 台のオーケストレータを設置し，負荷分散をそのコントローラ集団のみで行うなどの対策が考えられる．この場合，各オーケストレータをネットワーク全体のトポロジー情報の分散データベースとして利用することも出来る．大規模なネットワークのトポロジー情報の保存は大量の記憶域を消費するため，オーケストレータが情報の集約を行い，コントローラは必要最小限の情報だけを保持し，オーケストレータを介して必要な情報を取得する．

情報の同期：従来の制御では，コントローラは自身が管理するエリアの情報を収集し，自身のエリアのフローを制御した．しかし，特定のエリアの Packet_In の処理を複数のコントローラに分散させる都合上，分担を行うコントローラ間で攻撃対象エリアのトポロジー情報の同期を行うことが求められる．このタイミングのずれによって，同じ Packet_In メッセージに対する出力がコントローラ間で異なるものになることがある．例として，ホスト情報の同期ずれによって，宛先が不明であったり，古い位置情報を基に経路選択をする等が考えられる．この場合，不適切な制御が行われてしまい，通信のエラーや，ホストの位置確認のための ARP リクエストのような不要な出力が発生することになる．ただし，この場合は即座に再要求や ARP リプライが返ってくると考えられるため，その時点でトポロジー情報の同期が完了していれば，改めて正しい制御が行われる．

負荷：制御アプリケーションの実装によっては，Packet_In に対する応答として，複数の Flow_Mod メッセージなどが生成されることが考えられる．また，応答メッセージをブロードキャストする必要がある場合，そのメッセージは攻撃対象エリアのすべてのスイッチに対して個別に送信する必要がある．複数の応答パケットを攻撃対象コントローラが逐一受信し，解析することは負荷の原因となるため，応答メッセージのカプセル化などによる対策が考えられる．ブロードキャストの場合，宛先ブロードキャストであることを示すヘッダを，送信するメッセージに付与して出力することで，攻撃対象コントローラが受け取るデータ量を減らすことが出来る．また，複数の宛先に対する個別の応答メッセージが出力される場合，ヘッダに宛先となる datapath_Id と，メッセージ長の組を列挙し，応答メッセージを連結させる等の対処を行うことで，攻撃対象コントローラ側の処理を軽減する．

コントローラは自身が担当するエリアの制御を行いながら，攻撃対象エリアの Packet_In の処理を行うため，担当エリアのトラフィック量によって，コントローラ間で負荷が偏ることがある．この場合，オーケストレータが各コントローラの負荷の観測を基に，分配の比率を調整することによる対処が考えられる．

また，負荷分散モード中に，攻撃対象のコントローラの処理性能に余裕がある場合，Packet_In メッセージの一部を攻撃対象コントローラが直接処理を行うことで，よりスループットの向上が見込まれる．ただし，処理を行う Packet_In のレートを過負荷を与えない程度に調整しなければならないため，事前のテストや稼働時のパラメータの推移から，Packet_In の処理量と負荷の相関を計算しておくことが求められる．

複数のエリアで過負荷が発生した場合，これらの処理を全てのコントローラで分配すると，各コントローラが保持するトポロジー情報のサイズが大きくなってしまう．また，トポロジー情報の更新頻度も増えてしまうため，負荷が大きくなってしまう恐れがある．この場合，過負荷が発生しているコントローラ毎に，転送先となるコントローラグループを個別に設定することで，各コントローラが保持するネットワーク情報の量を抑えることが出来る．

他の手法との併用：提案手法ではデータプレーンの拡張を行わないため，データプレーン側での SDN-DoS に対するフィルタリングなどの対策手法と併用することが出来る．また，提案システムの構成は，階層型の分散制御方式の構成に近い．そのため，階層型において上位コントローラで行われていた処理の一部をオーケストレータが行うことで，階層型の分散制御の利点を得ることも出来る．例として，ネットワーク全体の情報を基に計算する制御ルーチンを，オーケストレータ上で行うように定義することで，コントローラが行う処理を軽減することができると考えられる．

この手法は，SDN-DoS 攻撃のような一定の期間過負荷が継続する状況を想定したものだが，負荷の計測手法によっては，トラフィックの瞬間的な急増に反応してしまう可能性がある．また，各コントローラの制御エリア内のトラフィック量に偏りがある場合，負荷分散モード時にコントローラ間で負荷の不均衡が発生する可能性がある．そのため，ElastiCon のような動的にコントローラの制御エリアを調整する手法と組み合わせることが考えられる．オーケストレータに各コントローラの制御エリアの調整を行う機能を実装し，通常制御時の各コントローラの負荷をなるべく均一に保つことで，より負荷分散モードの処理能力を向上させることが出来る．

8. 今後の課題

本論文では，複数のコントローラによる分散制御が行われる SDN 環境で，コントローラを標的とした SDN-DoS に対処するための，コントローラ間の負荷分散システムを提案した．SDN-DoS では，少数のスイッチからコントローラへの大量の処理要求が発生するため，オーケストレータがコントローラの負荷を監視して攻撃の発生を検知し，攻撃によって発生する制御メッセージを全コントローラへ分散させることで計算リソースを増加させ，攻撃に対する制御スループットを向上させた．

このシステムの性能は，コントローラの数や性能，Packet_In メッセージの到着レート，および制御アプリケーションが 1 つの Packet_In の処理に要する時間等の状態によって変化すると考えられる．負荷分散は稼働しているコントローラの数が多いほど効果的であると考えられるため，コントローラ数と Packet In のレートの影響について評価を行っていく．また，この手法はスループットを向上させることが目的だが，転送によるレイテンシやオーバーヘッドも発生するため，通常の制御と比較した際にこれらの要素がどれほど影響を与えるかを確認する．

また，オーケストレータによる転送先コントローラのグループ化や，モード移行のための最適な閾値を動的に検出する手法の導入など，より最適なシステムの仕様を検討する．今後は，他の SDN-DoS 対策やスケーラビリティを向上させる手法を参考にし，SDN-DoS への高い耐久性を持った包括的な SDN アーキテクチャの検討を行っていく．

参考文献

[1] Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N. and Shenker, S.: NOX: Towards an Operating System for Networks, *SIGCOMM Comput. Commun. Rev.*, Vol. 38, No. 3, pp. 105–110 (2008).

[2] : Trema - GitHub Pages, `http://trema.github.io/trema/`.

[3] Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., Hama, T. and Shenker, S.: Onix: A Distributed Control Platform for Large-scale Production Networks, *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, pp. 351–364 (2010).

[4] Tootoonchian, A. and Ganjali, Y.: HyperFlow: A Distributed Control Plane for OpenFlow, *Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking*, pp. 3–3 (2010).

[5] Karakus, M. and Durresi, A.: A survey: Control plane scalability issues and approaches in Software-Defined Networking (SDN), *Computer Networks*, Vol. 112, pp. 279 – 293 (2017).

[6] Berde, P., Gerola, M., Hart, J., Higuchi, Y., Kobayashi, M., Koide, T., Lantz, B., O'Connor, B., Radoslavov, P., Snow, W. and Parulkar, G.: ONOS: Towards an Open, Distributed SDN OS, *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, HotSDN '14, New York, NY, USA, ACM, pp. 1–6 (2014).

[7] Benson, T., Akella, A. and Maltz, D. A.: Network Traffic Characteristics of Data Centers in the Wild, *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, pp. 267–280 (2010).

[8] Li, W., Meng, W. and Kwok, L. F.: A Survey on OpenFlow-based Software Defined Networks, *J. Netw. Comput. Appl.*, Vol. 68, No. C, pp. 126–139 (2016).

[9] : 製品仕様: SDN 対応製品　UNIVERGE PF シリーズ — NEC, `http://jpn.nec.com/univerge/pflow/spec_pfs.html`.

[10] Tavakoli, A., Casado, M., Koponen, T. and Shenker, S.: Applying NOX to the Datacenter, *Proc. of workshop on Hot Topics in Networks (HotNets-VIII)* (2009).

[11] Cai, Z., Cox, A. L. and Ng, T. S. E.: Maestro: A System for Scalable OpenFlow Control, Technical report, Rice University (2011).

[12] Dixit, A., Hao, F., Mukherjee, S., Lakshman, T. V. and Kompella, R. R.: ElastiCon; an elastic distributed SDN controller, *2014 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, pp. 17–27 (2014).

[13] Bari, M. F., Roy, A. R., Chowdhury, S. R., Zhang, Q., Zhani, M. F., Ahmed, R. and Boutaba, R.: Dynamic Controller Provisioning in Software Defined Networks, *Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013)*, pp. 18–25 (2013).