Packet In メッセージに基づく SDN ネットワーク内の DDoS 攻撃検知

尤 翔[†1,2]　フォン ヤオカイ[†1,2]　櫻井幸一[†1,2]

概要：OpenFlow プロトコルを用いて，仮想化ネットワーク技術 SDN(Software Defined Network)は現在広く使われている．また近年 DDoS 攻撃の件数は年々増加してきた，SDN 内 DDoS 攻撃を検知するため，OpenFlow スイッチ中のフローテーブルに記録されたデータを分析する検知手法が多数提案されている．しかし，SDN ではネットワーク内の通信を集中管理するため，大量の処理負荷を与える DDoS（Distributed Denial of Service）攻撃を検知する際，OpenFlow コントローラの処理負荷を考えなければいけない．本稿では，コントローラの処理負荷を減少するため，フローテーブルのデータを収集せず，コントローラとスイッチ間の通信用の Packet In メッセージから四つの特徴を抽出し，リアルタイムに DDoS 攻撃の検知を行う．更に，既存研究でよく利用される検知時間間隔（パラメータの一つ）が決め難しいため，本研究はそれを回避して，検知する前にトリガーを追加することより，軽量かつ動的な DDoS 攻撃を検知することができる．

キーワード：SDN，DDoS 攻撃，OpenFlow プロトコル，OpenFlow コントローラ，OpenFlow スイッチ

Packet In message based DDoS attack detection in SDN network using OpenFlow

XIANG YOU[†1,2]　YAOKAI FENG[†1,2]
KOICHI SAKURAI[†1,2]

Abstract: Using the OpenFlow protocol, the virtual network technology SDN (Software Defined Network) is now widely used. In recent years, the number of DDoS attacks has been increasing year by year. To detect DDoS attacks in SDN, data recorded in the flow table in OpenFlow switch is analyzed and various detection methods are submitted. However, since SDN centrally manages communication within the network, when detecting DDoS (Distributed Denial of Service) attack which gives a large amount of processing load, the processing load of the OpenFlow controller must be considered. In this paper, in order to reduce the processing load of the controller, we do not collect data of the flow table, extract three features from the Packet In message for communication between the controller and the switch, and perform real-time attack detection. Furthermore, to avoid stiff detection time intervals, triggers will be added before detection to detect light and dynamic DDoS attacks.

Keywords: SDN , DDoS Attack , OpenFlow protocol , OpenFlow controller , OpenFlow Switch

1. はじめに

SDN とは，コンピュータネットワークを構成する通信機器を単一のソフトウェアによって集中的に制御し，ネットワークの構成や性能・機能を柔軟に，動的に変更できる技術である．

従来のネットワークでは通信機器の一台ずつが独立した OS や制御ソフト，経路選択機能をデータ転送機能有しており，設定や構成などの作業も一台ずつ，あるいは機器の種類毎に行う必要があり，ネットワークの構成は固定的だった．SDN の最大の特徴は「データ転送(Data Plane)」と「経路制御(Control Plane)」の機能を分離したアーキテクチャを採用していることにある．データ転送と経路制御の分離によって，複数のネットワーク機器をソフトウェアによって一カ所で集中管理することにより，どの機器にどのような動作をさせるか柔軟に設定することができる．その柔軟性のため, SDN は backbone ネットワーク, データセンター，アクセスネットワーク，ワイヤレスネットワークなどのアプリケーションで広く研究されている［1］.

新興のアーキテクチャとして，それ自身のセキュリティを確保する必要がある．D.Kreutz らは，SDN おけるセキュリティ課題について整理を行っており，七つの主な潜在的なセキュリティ問題が生じると指摘している［2］.

†1　九州大学，福岡県福岡市西区元岡 744 番地
Kyushu University, 744, Motooka, Nishi-ku, Fukuoka, 819- 0395, JAPAN.
†2　九州先端科学技術研究所，福岡県福岡市早良区百道浜 2 丁目 1 番 22 号福岡 SRP センタービル 7 階
Institute of Systems, Information Technologies and Nanotechnologies (ISIT),Fukuoka SRP Center Building
7F,2-1-22,Momochihama,Sawara-ku,Fukuoka,814-0001,JAPAN

① 偽装されたパケットの送信
② シィッチの脆弱性と突いた攻撃
③ コントロールプレーンへの攻撃
④ コントローラの脆弱性を突いた攻撃
⑤ コントローラとアプリケーション間の信頼性確保機構の欠落
⑥ 管理者マシンの脆弱性を突いた攻撃
⑦ 信頼を裏付ける情報取得機構の欠落

しかしながら，SDN が提供するネットワークプログラミング，ネットワーク監視，動的フローポリシーの実装を使用すると，ネットワークフォレンジックセキュリティポリシーの変更, セキュリティーサービスなどを SDN で実現できる．

そのセキュリティ問題の中で最も緊急かつ困難なセキュリティ問題の一つは Distributed Denial of Service(DDoS) 攻撃である．DDoS の攻撃手法となるアクセス自体はこれまでの DDoS 攻撃のアクセスとほとんど変わりがなく，防御や追跡することが困難であるため，深刻な被害を引き起こす可能性がある．例えば，米 DNS サービス大手の Dyn は 2016 年 10 月 21 日の午前 11 時ごろ，大規模な分散型サービス妨害(DDoS)攻撃を受けてダウンした. これにより，同サービスを使っている Twitter，Spotify，Reddit，Netflix，Wall Street Jounral など多くのサービスが主に米国で約 6 時間に終わって利用できなくなっていた［3］．したがって，DDoS 攻撃を効果的に検知することは，SDN における将来のネットワークアーキテクチャの展開にとって非常に重要である．

DDoS 攻撃からサーバを守る研究として，これまで，SDN における様々な DDoS 攻撃検知のメカニズムを既に提案されてきた．既存の検知方法はほとんど統計用時間間隔を事前に決める必要がある［4］［5］［6］［7］．しかしながら，適切な時間間隔を決めることは困難である．選択した期間が長すぎると，DDoS 攻撃を開始するから検知を始めるまでの応答時間が長くなり，コントローラとスイッチが非常に大量の攻撃パケットを処理し，コントローラとスイッチを破壊することも可能である．逆に，周期が短すぎると，DDoS 攻撃検知が頻繁に開始され，コントローラは多くのリソース(CPU とネットワーク帯域幅)を無駄にし，コントローラの効率に影響する．

この問題を解決した既存手法として，Software Defined Anti-DDoS(SD-Anti-DDoS)［1］が提出された. SD-Anti-DDoS は，固定的な周期を使わなく，DDoS 攻撃検知を開始する前，動的なトリガーを追加する．このトリガーとは，DDoS 攻撃が発生する時，コントローラが受信した packet-in メセージの異常増加に基づく，適切なタイミングに攻撃検知を開始することができる．しかし，この手法では，攻撃検知を行う際に，コントローラがすべてのスイッチから統計データを収集と処理することが必要である．したがって，集中制御プレーン（コントローラ）が過負荷になってしまうという問題が存在する．

上述の問題を解決するため，我々は SDN における DDoS 攻撃を検知するための新たなメカニズムを提案する．

本研究では，スイッチ中の統計データを収集せずに動的なトリガーを用いた DDoS 攻撃の検知手法を提案する．攻撃検知する際のみにフローテーブルのデータを収集することではなく，すべての時間帯に分散して Packet In メッセージから必要なデータを収集と処理することで，リアルタイム攻撃検知を行い，コントローラの処理負荷を減少することができる．更に，事前に検知時間間隔を決める必要がなくなり，軽量かつ動的な DDoS 攻撃を検知することができる．

2. 背景

2.1 SDN

図 1 に示すように，伝統的なインターネットアーキテクチャと SDN との主な違うところは, 後者が制御および転送面を切り離すことである．SDN では，制御機能は転送から切り離され, ソフトウェアに基づく SDN コントローラで集中管理される．

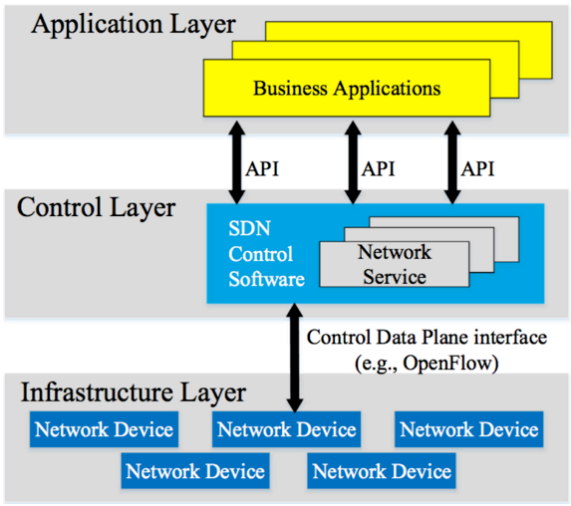

図 1　SDN のアーキテクチャ［1］

2.2 OpenFlow

OpenFlow は，SDN の制御層と Infrastructure 層を接続するための最初の通信プロトコルである［9］．図 2 に示すように，経路制御は OpenFlow コントローラ，データ転送は OpenFlow スイッチと呼ばれるものがそれぞれを担い，コントローラとスイッチは OpenFlow プロトコルで通信を行っている［8］．現在，OpenFlow プロトコルは SDN の事実上の標準となっている．この論文では，コントローラとスイッチを接続するための通信プロトコルとして OpenFlow v1.0 が選択されて，OpenFlow の異なるバージョン間でわずかなパフォーマンスの差が存在する可能性があ

る．

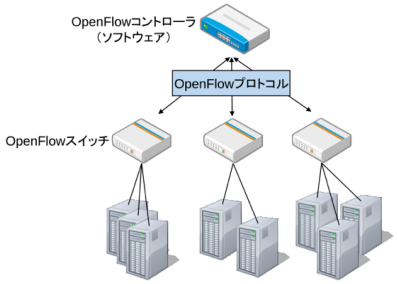

図２　OpenFlow コントローラとスイッチ［8］

2.3　OpenFlow スイッチ

OpenFlow スイッチは，Datapath ID と呼ばれる 64bit の識別子を持つ．コントローラ側から識別するために，ネットワーク中の各 OpenFlow スイッチには，それぞれ特別な Datapath ID が割り当てられている必要がある．

図３で示した通り，OpenFlow スイッチはパケットの lookup と転送機能を担うフローテーブルと，スイッチをコントローラに接続するための安全なチャネルで構成されている． フローテーブルでは，一連のフローエントリで構成され，各フローエントリには，ヘッダーフィールド，統計情報，およびアクションが含まれている．このフローエントリは，OpenFlow コントローラにより生成され，OpenFlow プロトコルを使って，OpenFlow スイッチへと送られる．送られたフローはスイッチ中のフローテーブルに格納される．

パケットが来る度に，そのフローエントリを参照してアクションを行う．パケットがフローテーブル内の既存のすべてのフローエントリと一致しない場合，コントローラに問い合わせて指示を仰ぐ．コントローラから受けた指示内容は，新たなフローエントリとして，フローテーブルに書き込まれる．

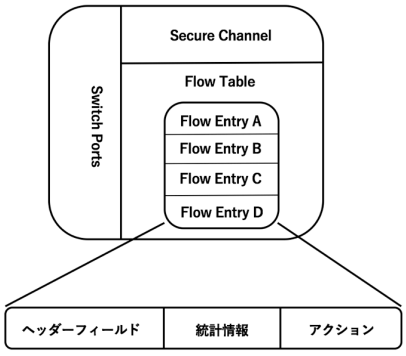

図３　OpenFlow スイッチのアーキテクチャ

2.4　フローエントリ

フローエントリは以下の三つの部分で構成されている．

① ヘッダーフィールド
② アクション
③ 統計情報

①ヘッダーフィールド

ヘッダフィールドとは，表 1 に示す 12 種類のフィールドで，受信したパケットを識別するための条件として用いられる．フロー識別に使用しないフィールドにはワイルドカードを指定することで，任意のフィールドのみを条件として用いることができる．例えば，“受信ポート番号が 1 であり，かつ宛先 MAC アドレスが FF:FF:FF:FF:FF:FF”や“IP パケットであり，その送信元 IP アドレスが 192.168.1.1”といった条件を指定することができる．

②アクション

アクションは，ヘッダフィールドにマッチしたパケットの処理を指定する．仕様で規定されているアクションを表 2 に示している．一つのフローに対して，複数のアクションを指定することも可能である．例えば，“宛先 IP アドレスを書き換えた上での指定のポートからの出力”や，“指定のポートから出力させた後，さらに別のポートからも出力”といった動作を指定できる．

③統計情報

OpenFlow ではフローエントリごとにカウンタを持っており，次の統計情報を取得できる．

- 受信パケット数
- 受信バイト数
- フローエントリが作られてからの経過時間

例えば「リモコン関連の問い合わせ数は 9 件」とマニュアルに記録したように,「このフローエントリにしたがって処理したパケット数は 80 個」と言った情報が入る．

表１　ヘッダーフィールド

フィールド	説明
Ingress port	受信ポート
Ethernet src address	送信元 MAC アドレス
Ethernet dst address	宛先 MAC アドレス
Ethernet type	プロトコル種別
VLAN id	VLAN ID
VLAN priority	VLAN PCP 値
IP src address	IP 送信元アドレス
IP dst address	IP 宛先アドレス
IP protocol number	プロトコル番号
IP ToS bits	ToS 値
Transport src port	送信元ポート番号
Transport dst port	宛先ポート番号

表 2　アクション

アクション	説明
Forward	パケットを転送する
Enqueue	指定のキューに入れる
Drop	パケットを破棄する
Modify-Field	フィールドを書き換える

2.5　OpenFlow の仕様

OpenFlow スイッチでは，OpenFlow コントローラから制御され，OpenFlow コントローラとスイッチとの間は，TCP/TLS(Transport Layer Security)を用いて接続される．この TCP/TLS コネクションを用いて，コントローラとスイッチ間の通信に用いられるのが OpenFlow プロトコルである．OpenFlow プロトコルには，それぞれのメッセージが定義されている．この中で主要なメッセージについて，次に説明を行う．

① Packet In メッセージ：フローテーブル中にマッチするフローがなかった場合，受信パケットをコントローラへと送るために用いられる．このメッセージで送られるパケットを元にフローを作成するなど，OpenFlow 特有の処理を実現するために欠かせないメッセージである．

② Packet Out メッセージ：Packet In でコントローラに送られてきたパケットを本来の宛先に送るために，スイッチ側へと送り返す際に用いられる．また，コントローラが独自に作ったパケットを出力させる際にも用いられる．

③ Flow Mod メッセージ：コントローラが作ったフローをスイッチへと送るために用いられる．その際に，フローを設定してから一定時間後に消すためのハードタイムアウトと，フローが参照されない時間が一定時間経過した後に消すためのアイドルタイムアウトの，2 種類のタイムアウト値を設定できる．

OpenFlow スイッチとコントローラは OpenFlow 仕様で規定されたメセージをやり取りしながら動作する．

上述のように，集中制御，プログラムを書いて複雑な処理ができなど，ネットワーク仮想化技術(SDN)の利点を利用すると，リアルタイム監視，正確な分析，迅速な対応などの技術をサポートする．

3.　先行研究

本章では，SDN における OpenFlow を用いた DDoS 攻撃検知の先行研究について紹介する．

既存研究の中，SDN における DDoS 攻撃検知のメカニズムは，パケット検査に基づくもの［6］［7］［10］［11］と，フローエントリ検査に基づくもの［1］［4］［5］を含む二つのタイプに分類している．

パケット検査に基づく DDoS 攻撃検知では，ネットワークが攻撃されていない場合でも，検知するために全部のパケットをチェックする必要がある．このプロセスにより，コントローラは通常のメッセージを処理する時，より多くの時間とリソースを費やしてしまう．パケット検査に基づく DDoS 攻撃検知と違う，フローエントリが当該フローに合致するパケットの統計情報を記録しているため，フローエントリ検査に基づく DDoS 攻撃検知では，すべてのパケットを調べるのではなく，スイッチ内のフローエントリを検査するだけで，DDoS 攻撃検知を行うことができる．したがって，フローエントリ検査に基づく DDoS 攻撃検知は，パケット検査に基づく DDoS 攻撃検知より，軽量で効率的である．

Braga らは，OpenFlow を用いてフローエントリに基づく DDoS flooding 攻撃検知システムを提案した［4］．図 4 に示すように，このシステムは三つのモジュールに分けられる．三つのモジュールの検知ループを以下に説明する．

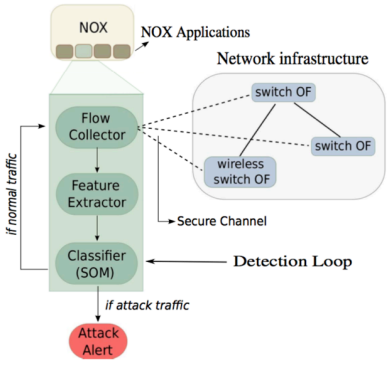

図 4　検知ループ

- Flow Collector モジュールでは，あらかじめ定めた時間間隔毎に，NOX コントローラに登録した OpenFlow スイッチから，すべてのフローテーブル内のフローエントリを収集する．
- Feature Extractor モジュールでは，収集されたフローエントリを受けて，DDoS flooding 攻撃検知に関する六つの特徴量を抽出し，それらの特徴量を分類器に渡す．
- Classifier モジュールでは，上述の六つの特徴量に基づく，SOM(Self Organizing Maps)で分析し，DDoS flooding 攻撃が発生しているかどうかを判断する．

この手法の利点として，パケット検査に基づく DDoS 検知手法と比べると，Braga らの手法はフローエントリに基づくため，より軽量である．

しかし，検知ループの間，適切な時間間隔を選択することは困難である．第一章に述べたように，長すぎるまたは短すぎると，効率が低下になってしまう．

そこで，Cui らは，Packet In トリガーという概念を提出した［1］．Packet In メッセージでは，きたパケットはフ

ローエントリに合致しない場合，コントローラに Packet In メッセージを送って問い合わせる．このトリガーは Packet In メッセージ基づく，DDoS 攻撃検知を行う前に，Packet In メッセージの異常増加があるかどうかを判断する．原理としては，IP Spoofing を使って DDoS 攻撃が発生する場合，スイッチが膨大な数の Packet_in メッセージが生成される．一方, OpenFlow スイッチはすべてのホストに合致するフローエントリを維持することができないため，DDoS 攻撃は IP Spoofing を使っていない場合でも，大量の新たな Packet In メッセージを生じる．

Packet In メッセージの特徴に基づいて, Cui らは Packet In メッセージの異常増加を検知する Packet In トリガーを提案した．Packet In トリガーを導入すると，DDoS 攻撃に対して，より迅速に対応しながら，コントローラとスイッチの負荷も軽減する．

しかし，本来コントローラはネットワークを制御しており，これにフローエントリの統計データ収集を加えて，特に検知する際，すべてのフローエントリを収集すると，コントローラに過負荷をかけてしまう．特に，高速 DDoS 攻撃が発生する場合, OpenFlow スイッチから生成された数千の Packet In メッセージがコントローラに送られ，負荷が更に大きくなって，コントローラに DoS 攻撃を引き起こす可能性がある［5］.

4. 提案手法

本論文では，Packet In トリガー［1］を使って，スイッチから統計情報を収集することがなく，Packet In メッセージに基づく DDoS 攻撃検知手法を提案する．

4.1 設計原理

これまで，SDN 内 DDoS 攻撃検知について，多くの研究は，より軽量，より効率的なフローエントリに基づく DDoS 攻撃検知に注目している．

しかし, それらの DDoS 攻撃検知は二つの問題がある．一つは，多くの検知手法では，周期的(あらかじめ定めた時間間隔)に DDoS 攻撃検知を行うことである．この周期が長すぎると，DDoS 攻撃を迅速に検知することができない．一方，短すぎると，検知が頻繁にされ，CPU や帯域幅などのリソースが無駄にして，効率が低くなる，適切な周期を選択することが難しい．この問題を解決するため，本論文では Packet In トリガーを採用することにより，適切なタイミングに DDoS 攻撃検知を行うことができる．

もう一つの問題は，既存のフローエントリに基づく DDoS 攻撃検知手法では，攻撃検知を行う際に，コントローラに登録したすべてのスイッチから統計情報を収集することを必要とする．しかし，そうすると，DDoS 攻撃が発生する場合，大量の Packet In メッセージが送られ，コントローラの処理負荷が増加しているうちに，大量の統計データ収集の処理が攻撃検知の時に集まっているため，コントローラが過負荷になる可能性がある．この問題を解決するため，本論文では，攻撃検知用データを収集することをすべての時間帯に分散し，より負荷は軽量の Packet In メッセージに基づく DDoS 攻撃検知手法を提案した．

原理としては，コントローラが受信した Packet In メッセージによって，Flow Mod を作り，Flow Mod メッセージをスイッチに送信することで，フローエントリを設定することができる．したがって，一つの Packet In メッセージは一つのフローエントリに合致している．すなわち，ヘダーフィールド部分の統計情報(IP アドレス，ポート番号など)は，フローエントリを収集しなくても，Packet In メッセージから得られる. 本論文では, Packet In メッセージにより，分散の時間帯にヘダーフィールド部分のデータを収集し，この部分のデータに基づいて，DDoS 攻撃検知を行う．したがって, より負荷は軽量の DDoS 攻撃検知を実現できる．

4.2 アーキテクチャ

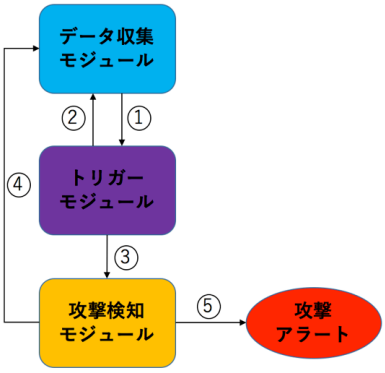

図５　DDoS 攻撃検知の状態図

図５に示すように，本手法では，三つのモジュールに分けられる．具体的なモジュール内容とモジュール間の遷移条件は次に説明する．

（1）データ収集モジュールとトリガーモジュール

図７に示すように，データ収集モジュールでは，きた Packet In メッセージを保存し，Packet In メッセージの数を記録している，m（パラメータ）個の Packet In メッセージ毎に，経過した時間間隔を記録し，以下の公式より，Packet In メッセージの到着速度を計算する．

- Packet In メッセージの到着速度：

$$V_{packet\text{-}in}=\frac{m}{current\ time-last\ time}$$

このモジュールは，DDoS 攻撃検知に利用された，以下の三つの特徴量を抽出する．

- dst IP のエントロピー：

$$h_{dstIP}= -\sum_{i=1}^{n} pi * \log_2 pi$$ (n は dstIP の種類)

- dstPort のエントロピー：

$h_{dstPort}= -\sum_{i=1}^{n} pi * \log_2 pi$（n は dstPort の種類）

- srcIP のエントロピー：

$h_{srcIP}= -\sum_{i=1}^{n} pi * \log_2 pi$（n は srcIP の種類）

図６の通り，計算で得られたデータを保存するため，node を作る．作った node は node list に入って，node list 中の node をすべて検査し，生成されたからの経過時間は time out 時間以上になると，node を削除して node list を更新する．処理完了の後に，状態遷移①が発生し，システムはトリガーモジュールにいく．

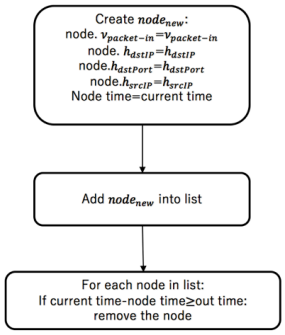

図６　node の生成と更新

（2）トリガーモジュール

トリガーモジュールでは，Packet In メッセージの特徴に基づいて，Packet In メッセージの異常バーストを検出するために利用されるものである．

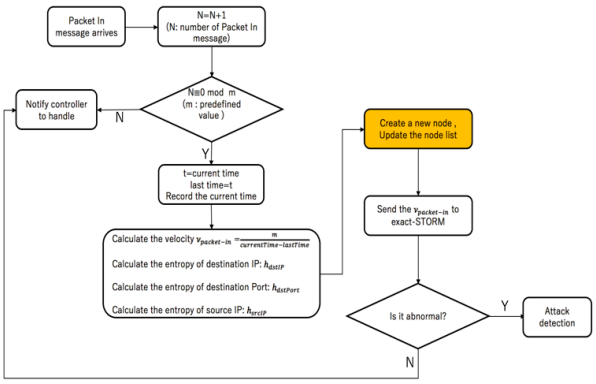

図７　データ収集とトリガーのワークフロー

このモジュールは，exact-STORM［12］というアルゴリズムを利用する．データ収集モジュールからもらった $V_{packet\text{-}in}$(Packet In メッセージの到着速度)はアルゴリズムに入力すると異常かどうかを判断する．

Exact-STORM は，$V_{packet\text{-}in}$を作られた node に保存して，node は ISB というデックス付きストリームバッファに入れる．具体的な内容は以下に説明する．

Method 1: update_data_stream(velocity)
Input: the velocity of packet in message
Output: nothing
Method 2: find_packet_in_burst()
Input: nothing
1: R=the predefined radius
2: **if** the oldest node in ISB expires then
3:　remove it from ISB
4: **end**
5: create a new node n(curr), with n(curr).obj=velocity,
　n(curr).id = id, n(curr).nn_before=[], n(curr).count_after=1
6: **for** each node in ISB do
7:　**if** abs(n(curr).obj-velocity)<R then
8:　　increase n(curr).count_after by one
9:　　append .id to n(curr).nn_before
10:　**end**
11: **end**
12: insert n(curr) to ISB
Output: whether there is an outlier in data stream
1: **for** each node n in ISB do
2:　prec_neighs=number of identifiers in n.nn_before
3:　succ_neighs=number of identifiers in n.count_after
4:　**if** (prec_neighs+succ_neighs)<k then
5:　　return True
6:　**end**
7: **end**

Exact-STORM アルゴリズムで，異常があるかどうかを判断すると，システム状態の遷移を決める．異常がない場合，システムがデータ収集モジュールに戻る(遷移②)．異常がある場合，システムが攻撃検知モジュールへいく(遷移③)．

(3)攻撃検知モジュール

あるネットワークについて，ネットワーク中のホストに対して，よく接続する相手(ホスト，サーバなど)がある．一方では，ほぼ接続しないホストやサーバも存在する．したがって，正常のネットワーク通信について，常態が存在する．

その常態に基づいて，Packet In メッセージから抽出した三つの特徴量(dst IP のエントロピー，dstPort のエントロピー，srcIP のエントロピー)は，正規分布に従うことを仮定する．

検知方法として，このモジュールは，エントロピーに基づく方法を利用する．この方法は，ネットワークトポロジーやトラフィック特性と独立し，様々な種類のネットワーク監視に適用できる．

エントロピーでは，収集したデータのランダム性を推定することができ，高いエントロピーはより分散された確

率分布を意味し，低いエントロピーはより集中された確率分布を意味する．

DDoS 攻撃が発生する場合，標的となるコンピュータに対して複数のマシンから大量の処理負荷を与えるため，dst IP のエントロピーと dstPort のエントロピーが明らかに減少し，srcIP のエントロピーも正常より増えることを引き起こすことがわかる．

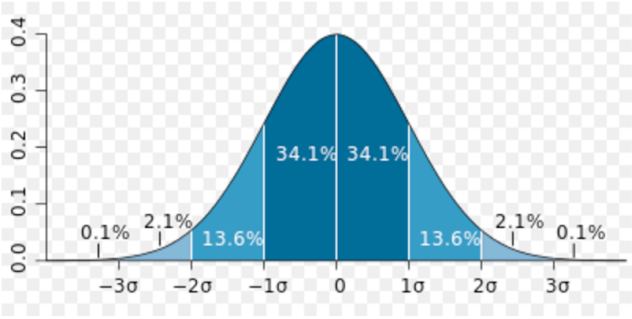

図８　(画像引用元：正規分布-wikipedia)

- μ ：平均
- σ ：標準偏差

エントロピーに基づく DDoS 攻撃を検知するため，このモジュールは正規分布を利用する．図８に示した通り，正規分布では以下の特性がある．

- $\mu \pm \sigma$ の範囲に全体の約 68.3%が含まれる
- $\mu \pm 2\sigma$ の範囲に全体の約 95.5%が含まれる
- $\mu \pm 3\sigma$ の範囲に全体の約 99.7%が含まれる

この性質に基づいて，4.2（１）に述べた node list から，関連データ$(node.h_{dstIP}, node.h_{dstPort}, node.h_{srcIP})$を取得し，抽出された三つの特徴量は，以下の三つの条件を満たす場合，攻撃検知モジュールは，DDoS 攻撃が発生していると判断し，システムに通知する(遷移⑤).

- 条件一，dstIP のエントロピー(h_{dstIP})について：

$$h_{dstIP} < \mu_{node.h_{dstIP}} - 2^{*}\sigma_{node.h_{dstIP}}$$

- 条件二，dstPort のエントロピー$(h_{dstPort})$について：

$$h_{dstPort} < \mu_{node.h_{dstPort}} - 2^{*}\sigma_{node.h_{dstPort}}$$

- 条件三，srcIP のエントロピー(h_{srcIP})について：

$$h_{srcIP} > \mu_{node.h_{srcIP}} + \sigma_{node.h_{srcIP}}$$

そうではない場合，DDoS 攻撃が発生していないと判断することで，システムはデータ収集モジュールに戻る(遷移④).

5. 実験

本章では，上述の Packet In に基づく DDoS 攻撃検知システムを含むコントローラを作成し，攻撃検知を実験した．対照実験することで，フローエントリに基づく DDoS 攻撃検知の性能と比較することができた．

5.1 実験設計

提案された Packet In に基づく DDoS 攻撃検知システムは，Ryu コントローラ［13］を利用することで実現した．このメカニズムをシミュレートするために，Mininet［14］という実験的なプラットフォームが使用された．

Mininet とは，ネットワークエミュレータで，一つの Linux カーネル上で，複数のホスト，スイッチ，ルーターを組み合わせて仮想的にネットワークを構築することができる．本実験は Mininet で，コアスイッチ（c1），アグリゲーションスイッチ（a2-a5），エッジスイッチ（e6-e25），100 台のホスト（h1-h100）があり，25 台のスイッチを含むネットワークを作成した．

正常な通信を模倣するため，我々は Mininet にカスタムコマンドを追加することで，各のホスト(ホスト番号は q)毎に確率 Pt，Pa，Pc で，ホスト番号は(q+i)，(q+j)，(q+k)のホストにパケットを送るという確率モジュールのネットワークを作ることができた．

本実験の DDoS 攻撃トラフィックは TFN2K によって生成された．TFN2K は，よく知られている DDoS 攻撃ツールである．我々は,20 台のホストを踏み台として，TFN2K で，IP Spoofing を使用して標的ホストに UDP flooding という DDoS 攻撃を実行した．重要なパラメータは表 4 に設定されている．

表 3　UDP flooding 検知実験で用いたパラメータ

モジュール	パラメータ	定義	値
データ収集	m	m 個 Packet In メッセージ毎に，特徴量を抽出する	1000
トリガー	R	exact-STORM の radius	120
	s	ISB のサイズ	40
	k	exact-STORM 中，隣の node 数の最小値	20

5.2 実験結果

Packet In メッセージに基づく DDoS 攻撃検知の結果は図 9 に示した．攻撃がされた 20 分前に，データ記録の起点として，図 9 中は 0 分を書いている．正常通信の場合，dstIP のエントロピーは約 29bit, dstPort は約 28bit, srcIP は約 28bit である．起点から 20 分に UDP Flooding 攻撃が開始され，

dstIP エントロピーと dstPort エントロピーが明らかに減少し，srcIP エントロピーが増加していくと，4.2（３）の検知条件を満たして攻撃が検知された．

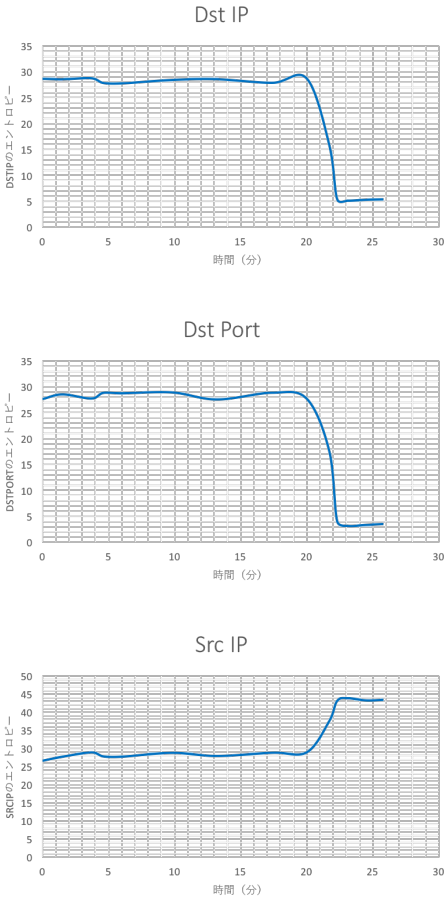

図 9　エントロピー

表５に示すように，正常通信の場合，フローエントリに基づく攻撃検知について，コントローラの CPU 使用率は，Packet In メッセージに基づく攻撃検知より低い（5.8%< 11.3%）．原因としては，フローエントリに基づく攻撃検知が検知する時しか統計データを収集しない．一方，Packet In メッセージに基づく攻撃検知のデータ収集は，集中ではなく，すべての時間帯に分散してデータを集中する．

DDoS 攻撃検知を行う場合，フローエントリに基づく検知では，スイッチからすべてのフローエントリを収集することが必要となるため，コントローラの CPU 使用率は，明らかに Packet In メッセーに基づく攻撃検知より高い (63.4%>32.6%).

表 4　コントローラの CPU 使用率

攻撃検知方法	正常通信の場合	DDoS 攻撃検知の場合
フローエントリに基づく	5.8%	63.4%
Packet In に基づく	11.3%	32.6%

6. 結論

本提案手法を用いて，DDoS 攻撃に対して，srcIP が減少し，dstIP エントロピーと dstPort エントロピーが増加することを示した．従来のフローエントリに基づく DDoS 攻撃手法と比較すると，統計データの収集と予め処理（特徴量の抽出）は分散されるため，コントローラに対して，本提案手法の処理負荷が明らかに減少している．更に動的な Packet In トリガーを追加していることで，レスポンス時間は多くの既存研究より早い．

しかし，本論文では，正常通信を確率モジュールで模倣するため，実験データの精確性を確認することが難しい．本物のネットワークトラフィックを導入することで，精確のデータを得るのは必要である．また，攻撃元を追跡して DDoS 攻撃からの被害を減少し，完全な DDoS 攻撃防御システムを作ることが今後の課題である．

参考文献

[1] Y. Cui, L. Yan, S. Li, H. Xing, W. Pan, J. Zhu, et al. SD-Anti-DDoS: fast and efficient DDoS defense in software-defined networks J. Netw. Comput. Appl., 68 (2016), pp. 65–79.
[2] D.Kreutz, F.M.V.Ramos, P.Verissimo, “Towards Secure and Dependable Software-Defined Net- works”, Proc. of the second workshop on Hot topics in software defined networking (HotSDN’12), pp.55-60, August 2013.
[3] http://www.itmedia.co.jp/news/articles/1610/22/news024.html.
[4] R. Braga, E. Mota, A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow", Proc. IEEE 35th Conf. Local Comput. Netw., pp. 408-415.
[5] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, V. Maglaris Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments Comput. Netw., 62 (2014), pp. 122–136.
[6] S.A. Mehdi, J. Khalid, S.A. Khayam Revisiting traffic anomaly detection using software defined networking Recent Adv. Intrusion Detect. (2011), pp. 161–180.
[7] Wang, B., Zheng, Y., Lou, W., 2015. Ddos attack protection in the era of cloud computing and software-defined networking. Comput. Netw. 81 (April (22)), 308–319.
[8] 宮崎 亮輔, 川本 淳平, 松本 晋一, 櫻井 幸一, “攻撃検知のための端末非依存型システムを実現する OpenFlow コントローラの実装と評価”火の国情報シンポジウム, 2015.
[9] Nick McKeown et al., "OpenFlow: enabling innovation in campus networks," ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, pp. 69-74 , April 2008.
[10] Xiao, Peng, Qu, Wenyu, Qu, Heng, Li, Zhiyang, 2015. Detecting ddos attacks against data center with correlation analysis. Comput. Commun. 67 (August (1)), 66–74.
[11] Miao, R., Yu, M., Jain, N., 2014. Nimbus: cloud-scale attack detection and mitigation. In: Proceedings of SIGCOMM, Chicago, IL, USA, ACM, pp. 121–122.
[12] Angiulli, F., Fassetti, F., 2003. Detecting distance-based outliers in streams of data. In: Proceedings of the Sixteenth ACM Conference on Information and Knowl- edge Management, ACM, pp. 811–820.
[13] http://osrg.github.io/ryu/.
[14] http://mininet.org/.