リアルタイムバースト検出手法の利用による即応性を考慮したDDoS攻撃検知手法の検討

臼崎 翔太郎[1] 有川 佑樹[1] 山場 久昭[1] 油田 健太郎[1] 久保田 真一郎[2] 岡崎 直宣[1]

概要：DDoS 攻撃を検知する統計的検知手法では，窓幅を大きくすることで検知精度を向上させるが，即応性に欠けるという問題点がある．一方，データストリームのバーストをリアルタイムに検出する手法ではイベント発生時に処理を行うことによりリアルタイムに解析が可能であり，さらに情報圧縮処理によって大量のイベント発生にも効率よく対応可能という利点がある．そこで我々はリアルタイムな解析を可能とし大量のイベント発生に強いリアルタイムバースト検出手法を利用し，即応性の高い DDoS 攻撃検知手法について検討する．本研究では，提案手法の有効性を確認するために評価実験を行い，検知精度と処理性能について議論する．

Investigation of Fast-Responsive DDoS Detection by Using Real-Time Burst Detection Method

SHOTARO USUZAKI[1] YUKI ARIKAWA[1] HISAAKI YAMABA[1] KENTARO ABURADA[1]
SHIN-ICHIRO KUBOTA[2] NAONOBU OKAZAKI[1]

1. はじめに

インターネットが社会基盤となっている現代社会ではDDoS(Distributed Denial-of-Service) 攻撃による被害は大きな脅威となっており，効果的な攻撃検知システムが望まれている．DDoS 攻撃はサイバー攻撃の一種であり，不正なトラフィックを攻撃対象のサーバに送り付けることによってユーザへのサービスを不能にさせる攻撃である．2000 年には大手の検索サイト yahoo に対する攻撃 [1] や，DNS のルートサーバに対する攻撃 [2] も報告されており，DDoS 攻撃による被害は大きな脅威となっていたが，2014 年には約 421Gbps の規模の攻撃 [3]，2016 年には IoT 機器からの 620Gbps の攻撃 [4] も確認されていることから，被害がますます大きくなっていくことが予想される．このような状況から，効果的な攻撃検知システムの開発が必要となってきている．

DDoS 攻撃の検知システムは，一般的にシグネチャ型とアノマリ型の二つに大別されるが，ともに即応性に欠けるという問題点が指摘されている [5]．シグネチャ型は，攻撃パケットのパターンをあらかじめ署名と呼ばれる形式でデータベースに保存しておき，パケットが到着するたびに，その特徴をデータベースと比較することによって攻撃を検知する手法である．この手法の利点として，署名の登録が容易であること，計算量が少ないこと，データベースに保存されている既知の攻撃に対しては検知精度が高いことが挙げられる [6]．しかしながら，登録した攻撃のパターンが増加するとパターンマッチングの計算量が多くなり，即応性が低下してしまうという問題点がある [5]．また，未知の攻撃に対しては誤検知が多くなってしまう．一方，アノマリ型は統計情報を用いて検知する手法である．アノマリ型では，あるパケット系列の統計情報を，正常時のものと比較することで攻撃か否かを判定する．比較対象となるパケットの系列（以降，窓幅と呼ぶ）は通常，時間かパケット数を単位とする．シグネチャ型と違って，アノマリ型は未知の攻撃に対しても誤検知率を下げることができるという利点がある．しかしながら，この手法では決定した窓幅を超えるまで検知ができないことから，窓幅を大きくすると即応性に欠けてしまう [5]．

[1] 宮崎大学工学部
[2] 熊本大学総合情報統括センター

本研究では，バースト検出手法を利用した，即応性の高い DDoS 攻撃検知手法について検討する．バースト検出手法とは，データマイニングの分野においてデータストリームの特徴を解析するための手法である．そのなかでも蛯名らの手法 [7] では，イベント発生ごとにバーストを解析するため，一定時間ごとに解析する手法に比べて素早くバーストを検出でき，イベントが発生しない際の無駄な計算を削減できるという利点がある．また，短時間に大量のイベントが発生した場合に負荷が大きくならないよう，ある期間に集中したイベントの情報をひとつに圧縮するので，大量のイベントが発生しても効率よく処理を行うことができる．蛯名らの手法はリアルタイムな解析が可能であり，大量のイベント発生に強いため，即応性の高い DDoS 攻撃検知システムとすることが期待できる．

以下，本稿の構成を述べる．2 章では関連研究を紹介し，問題点を指摘する．3 章ではリアルタイムバースト検出手法とそのデータ構造について説明し，DDoS 攻撃検知に有効な利点を解説する．4 章では提案手法について，攻撃判定式と追加した処理について述べる．5 章では実験環境，利用したデータセット，パラメータ値の設定について解説し，本手法の有効性を確認するため評価実験を行った結果を示す．6 章ではまとめと今後の課題について述べる．

2. 関連研究

DDoS 攻撃検知システムはシグネチャ型とアノマリ型に大別される．前者として [8]，後者として [9]，[10]，[11] を紹介する．

シグネチャ型の代表的なツールに Snort[8] がある．Snort の署名には，宛先 IP アドレス，送信元 IP アドレス，ポート番号などのヘッダー情報，そしてペイロード，メタデータなどのオプションが含まれる．パケットが到着すると，あらかじめ登録された署名と特徴を比較し攻撃を検知する．シグネチャ型検知手法は，常に署名がアップデートされている間は既存の攻撃に対して高い検知精度が期待される．また，署名の追加・管理が容易であるという利点がある [12]．欠点は方法やパラメータを変更した既存攻撃や未知の攻撃に対応できないという点，署名が大量に追加されるとパターンマッチングの計算量が多くなり高速計算性が失われる可能性がある点である．

文献 [9] は統計情報としてエントロピー値を利用するエントロピー手法の一種である．パケットのヘッダーフィールドの項目である IP アドレス，ポート番号などの情報源シンボルを確率変数として利用し，固定の窓幅 W（ここではパケット数）ごとにエントロピー値を計算する．DDoS 攻撃時などパケットが集中する際には，宛先 IP アドレス，宛先ポート番号等の情報量が低下することを利用し，攻撃検知を行う．エントロピー手法の利点としては，攻撃者がターゲット組織の通常時のエントロピー値を知ることは難しいため閾値をぎりぎり超えない攻撃が困難であること，エントロピー値の計算はカウントが主なのでパターンマッチング手法に比べて高速計算性があることが挙げられる．しかしながら，エントロピー手法では検知精度の向上のために窓幅を広くとる必要がある．なぜならば，窓幅が狭いとエントロピー値の揺らぎが大きくなってしまい，閾値による検知が困難になるためである．文献 [9] では誤検知率を下げるためには数万パケットの窓幅が必要であるとしており，即応性が高いとは言えない．

文献 [10] は SYN Flood 攻撃の検知手法である．通常時の SYN パケット到着レートが正規分布に従い，異常時には従わなくなることから，一定時間ごとにパケットをサンプリングして平均二乗誤差を計算し，正規分布との乖離度で検知を行う．この手法では作為的に送信されたパケットの到着を検知するため，攻撃パケット量が少ない場合でも効果的な検知が行える．しかしながら，正規分布を用いた平均二乗誤差の計算にオーバーヘッドがあり，高速計算性に欠けてしまう．この課題に対しては，あらかじめテーブルなどで正規分布の情報を保存しておくことで計算量の軽減が可能だとされている．

文献 [11] は SDN(Software Defined Networks) 環境における DDoS 攻撃緩和手法である．パケット収集，攻撃検知，攻撃緩和の 3 つのモジュールで構成されており，攻撃検知モジュールにはエントロピー手法が適用されている．攻撃検知モジュールでは，パケット収集モジュールによってサンプリングしたパケットデータからエントロピー値を計算し攻撃検知を行う．情報源シンボルは，宛先/送信元 IP アドレス，宛先/送信元ポート番号である．この手法では，各情報源のエントロピー値の変動の組み合わせにより DDoS 攻撃だけでなくワームやポートスキャンの攻撃も検知可能であるとしている．パケット収集モジュールでの窓幅は時間を単位としており，30s に設定している．この手法ではパケット収集を行った後に検知処理を行うため，少なくとも決定した窓幅を超えるまで検知を行うことができないという問題点がある．

3. リアルタイムバースト検出手法

バースト検出手法はデータストリームの異常状態を解析する手法であり，データマイニングの分野で利用されている．データストリームはオンラインニュースやブログ，電子掲示板といった高速に流れるデータのことを指す．データストリームではイベントの集中発生状態のことをバーストと呼び，これをいちはやく検出することは膨大に流れるデータから現在注目されている事柄を抽出することに繋がる．バースト検出手法はブログ解析，検索など，様々な分野で応用研究がなされている．本研究で利用する蝦名らの手法 [7]（以降，リアルタイムバースト検出手法と呼ぶ）は即応性の高いバースト検出手法で，以下の利点がある．

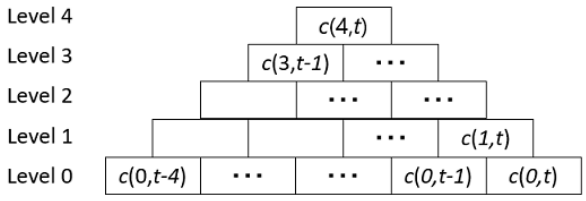

図 1　$n = 5$ の場合の Aggregation Pyramid

リアルタイムな解析が可能

イベントの発生ごとに処理を行うため，一定時間ごとに解析する手法に比べてバーストの発生にすぐに対応できる．またイベントが発生した際に初めて更新処理を行うため，イベントが発生しない時の無駄な計算を削減できる．

大量のデータにも対応可能

最小の窓幅として W_{min} を設定し，この時間に到着した分のイベントの情報を圧縮することにより，大量のイベントが発生しても効率よく処理を行うことができる．

これは，イベントの発生毎に処理を行うと，短時間に大量のイベントが発生した場合には処理負荷が大きくなってしまうためである．

3.1　データ構造

リアルタイムバースト検出手法で利用するデータ構造は，Zhang らの手法で提案された Aggregation pyramid[13] というデータ構造を参考としている．本節では，Zhang らの手法で提案された Aggregation Pyramid のデータ構造について述べた後，その構造を応用した，リアルタイムバースト検出手法で利用されるデータ構造について説明する．

3.1.1　Aggregation Pyramid

Aggregation Pyramid は，図 1 に示すように複数のセルで構成されており，n の階層を持っている．レベル h には $n - h$ 個のセルが存在しており，生成されたセルは各階層の右側に追加されていく．ここでセルの終了時間を t とすると，レベル h のセルは $c(h,t)$ と表現され，各セルはイベントの発生数，開始時刻，終了時刻，期間の長さの 4 つのイベント情報を保持する．

イベントが発生すると，生のイベント情報を保持するレベル 0 のセル $c(0,t)$ を生成する．次に $c(0,t-h)$ から $c(0,t)$ までのイベント情報を保持するレベル h のセル $c(h,t)$ を生成する．$c(h,t)$ は，$c(h-1,t-1)$ と $c(0,t)$ のイベント情報を集約することで生成される (図 2)．

3.1.2　リアルタイムバースト検出手法のデータ構造

リアルタイムバースト検出手法は Aggregation pyramid を応用したデータ構造を利用している．Zhang らの手法では各セルはイベントの発生数，開始時間，終了時間，期間の長さを保持していたが，リアルタイムバースト検出手法ではその代わりに合計到着間隔 $G(c(h,t))$，到着時刻

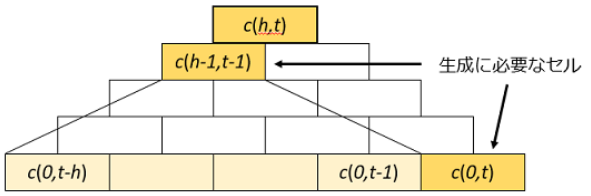

図 2　セルの集約の例

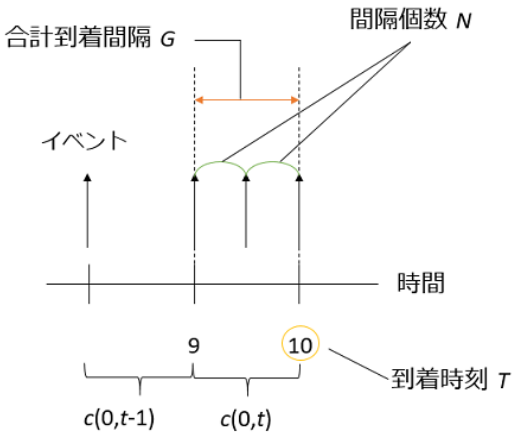

図 3　リアルタイムバースト検出手法のセルが保持するイベント情報

$T(c(h,t))$，間隔個数 $N(c(h,t))$ の 3 つのイベント情報を保持する (図 3)．

3.1.3　リアルタイムバースト検出手法のセル生成方法

リアルタイムバースト検出手法におけるデータ構造の構築過程を説明する．ここで $n+1$ 個の一連のイベントが発生したときの間隔の時間を $\boldsymbol{x} = (x_1, x_2, ...x_n)$ と表現する．新しくイベントが発生すると以下のルール [14] に従ってセルを生成し，データ構造を構築していく (図 4)．

(1) レベル 0 セルの生成方法

(a) $x_i \geq W_{min}$ の場合

- $G(c(0,t)) = x_i$
- $T(c(0,t)) = i+1$ 番目のイベント発生時刻
- $N(c(0,t)) = 1$
- $i = i+1$

(b) $x_i < W_{min}$ の場合

- $T(c(0,t)) = T(c(0,t-1)) + W_{min}$
- $N(c(0,t)) = T(c(0,t-1))$ から $T(c(0,t))$ 直前までの期間に発生したイベント発生数
- もし $T(c(0,t))$ にイベントが発生していないなら，$N(c(h,t)) = N(c(h,t)) - 1$
- $G(c(0,t)) = W_{min}$
- $i = i + N(c(0,t))$
- $x_i = T(c(0,t))$ から次のイベントが発生するまでの経過時間
- もし $c(0,t)$ で複数のイベントが発生しているなら，$x_i = 0$

(2) レベル h セルの生成方法

- $G(c(h,t)) = G(c(h-1,t-1)) + G(c(0,t))$
- $T(c(h,t)) = T(c(0,t))$
- $N(c(h,t)) = N(c(h-1,t-1)) + N(c(0,t))$
- 最上位セル $(h = n-1)$ のとき，$t = t+1$

このように新しくイベントが発生した時にのみデータの更新を行うため，定期的に更新処理を行うような手法に比べ，イベントが発生していない時の無駄な更新処理を削減できる．

3.2 バースト判定処理

バースト判定処理ではセル $c(h,t)$ が生成されると，これと集約している期間が重複していない直近の最上位レベル $(n-1)$ のセル $c(n-1,t-1-h)$(以降 $tgcell$ と呼ぶ) との間でそれぞれのイベント平均到着間隔を比較する．図 5 は各セルに対応する $tgcell$ を示したもので，同じ色で示されているセルが，対応する $tgcell$ となる．イベント平均到着間隔とはセルで集約しているイベントの到着間隔の平均値であり，セル $c(h,t)$ のイベント平均到着間隔は，セル $c(h,t)$ の合計到着間隔 $G(c(h,t))$ と間隔個数 $N(c(h,t))$ から式 (1) によって求められる．また，セル $c(h,t)$ のバーストの強さを表現するバースト性 [14] を式 (2) と定義する．

$$avg(c(h,t)) = \frac{G(c(h,t))}{N(c(h,t))} \tag{1}$$

$$brt(c(h,t)) = \frac{avg(c(h,t))}{avg(c(n-1,t-1-h))} \tag{2}$$

そして，バーストを判定するパラメータ $\beta(0 < \beta < 1)$ を設定し，閾値として式 (3) が満たされる時，バーストと判断する．

$$brt(c(h,t)) \leq \beta \tag{3}$$

この判定処理はイベントが発生した時に行われるので，バーストのリアルタイムな解析が可能となる．

次に過剰なバースト検出を抑制するためにパラメータ A_{min} を設定する．リアルタイムバースト検出手法ではパケット到着間隔の変化した割合で検知を行うが，これではパケット数が少ないときでもパケット到着間隔が大きく変化していれば攻撃であると判定してしまう．過剰な攻撃判定を防ぐため，間隔個数が十分に多い時に判定処理を行うようにする．具体的には，バースト判定処理は $N(c(h,t)) \geq A_{min}$ を満たした場合に行うようにする．

4. 提案手法

本研究ではリアルタイムバースト検出手法の「リアルタイムな解析が可能」,「大量のデータにも効率よく対応可能」という利点に着目し，パケットの到着を監視イベントとすることで DDoS 攻撃の検知に利用する．これによって，即応性と処理効率の高い攻撃検知が可能となる．ここからはイベント平均到着間隔をパケット平均到着間隔と呼ぶことにする．

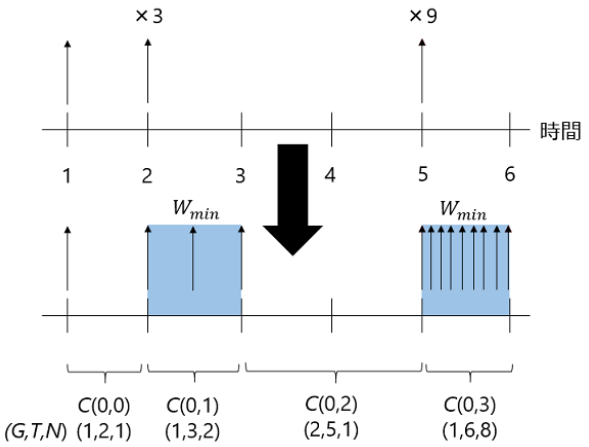

図 4 リアルタイムバースト検出手法におけるセルの生成方法

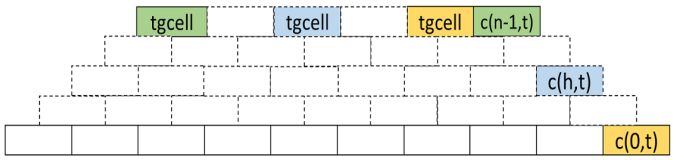

図 5 バースト判定処理における比較対象のセル（$tgcell$）

以前の研究 [15] では，リアルタイムバースト検出手法に従って式 (3) によってバーストと判定される時に攻撃を受けていると判断していた．しかしこれでは，長時間継続する同一レートの攻撃に対応できない，過剰に攻撃判定処理を行うという問題点がある．ここからは本研究で変更した箇所について説明する．

4.1 攻撃継続判定の追加

式 (3) を用いた攻撃検知手法は短時間の攻撃を検知することには向いているが，同一レートの攻撃が長時間継続した場合には，途中で攻撃が検知できなくなってしまう．なぜならば，攻撃が継続した際には直前のセルと現在のセルのパケット平均到着間隔の差が次第に小さくなってしまい，間隔の差が見られなくなってしまうためである．攻撃によって大量のパケットが到着しているにも関わらず検知できなくなる状況は，甚大な被害を引き起こすこととなるため，対策が必要となる．

長時間継続する同一レートの攻撃を検知する方法として，パケット到着間隔の差の広がりを監視する処理を追加することが考えられる．この攻撃ではパケット到着間隔の差が小さくなることが予想されるため，逆に間隔が広がった時に攻撃が終了したと判定するようにすれば，攻撃の継続的な検知が見込める．

そこで本研究では，攻撃開始を判定する攻撃開始判定状態，攻撃継続を判定する攻撃継続判定状態の 2 つの状態を定義し，それぞれ別の方法で攻撃の開始と継続とを判定さ

せるようにした．提案手法は常にどちらか一方の状態になるようにし，攻撃を受けていない時が攻撃開始判定状態，それ以外の時が攻撃継続判定状態である．攻撃開始判定状態では，式 (3) によって攻撃を受けていると判定されると，攻撃継続判定状態に遷移する．攻撃継続判定状態では，攻撃が継続していることを，式 (4) を満たした時に判定する．満たさない場合は攻撃が終了したと判断し，$T(c(0,t-1))$ を攻撃終了時刻として出力した後，攻撃開始判定状態に遷移する．

$$\frac{avg(c(0,t))}{avg(c(0,t-1))} \leq \frac{1}{brt(begin)} \tag{4}$$

閾値として，攻撃開始を検知した際のバースト性 $brt(begin)$ の逆数を設定している．これは，通常時のパケット流入量はほとんど定常だと考えられるので，攻撃開始時にパケット平均到着間隔が直前の間隔の $1/x$ 倍となった場合には，逆に x 倍間隔が広がって通常時のパケット平均到着間隔程度の大きさになった時に，攻撃が終了したと判定するのが適当だと考えたためである．例えば，攻撃開始を検知した際のパケット平均到着間隔が，直前の間隔の 1/100 倍となっていた場合，攻撃継続判定処理時ではパケット平均到着間隔が直前の間隔の 100 倍になったときに攻撃が終了したと判断する．攻撃継続判定において，比較対象のセルは n 個前までのセル情報を集約した $tgcell$ ではなく，直近のレベル 0 セルである．これは継続を判定する場合には遠い過去の情報を必要としないためである．

また，攻撃継続判定の追加は攻撃の終了をすばやく検知することにも繋がると考えられる．異常トラフィックに対するネットワーク管理プロセスにおいて，いちはやく攻撃終了を検知する重要性が文献 [16] で示されている．その目的は，異常が終息するまでの時間から原因調査の優先度を決定するため，あるいは正規のユーザに影響を与えないよう実施中の規制を速やかに解除するためとされている．提案手法では，パケット到着時に式 (4) によって間隔の広がりを監視し，攻撃の終了を検知する．攻撃時と通常時でパケット流入量に違いが見られる場合は，攻撃終了時にパケット到着間隔が大きく広がるため，すぐに攻撃の終了を判定することができると考えられる．

本研究では，攻撃開始を検知した時刻から攻撃継続の終了を検知した時刻までを攻撃判定期間とする．

なお，攻撃開始と判定した直後のセルで攻撃終了と判定された際には攻撃と判定しなかった．これは突発的に発生したパケットの集中到着による影響であると考えたためである．

4.2 過剰な攻撃判定の改善

以前の手法 [15] では，A_{min} を一定の値に設定していたが，高いレベルのセルになると集約しているセル数が多くなることから間隔個数が増加し，異常時でなくとも

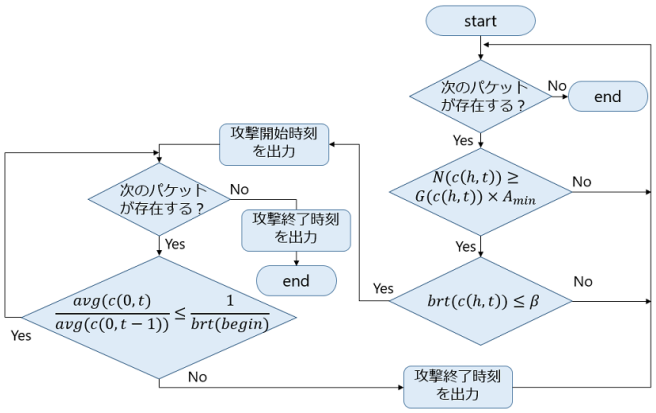

図 6 提案手法の攻撃検知の流れ

```
1: for t do
2:    h = 0
3:    while h < n do
4:       if 攻撃を受けていない then //攻撃開始判定状態
5:          if N(c(h,t)) ≥ G(c(h,t)) × A_min かつ brt(c(h,t)) ≤ β
            then
6:             T(c(h,t)) を攻撃開始時刻として出力
7:             攻撃継続判定状態に遷移
8:             break
9:          end if
10:      else    //攻撃継続判定状態
11:         if h = 0 かつ avg(c(0,t))/avg(c(0,t-1)) ≤ 1/brt(begin) then
12:            T(c(0,t-1)) を攻撃終了時刻として出力
13:            攻撃開始判定状態に遷移
14:            break
15:         end if
16:      end if
17:      h++
18:   end while
19: end for
```

図 7 提案手法の攻撃検知アルゴリズム

$N(c(h,t)) \geq A_{min}$ の式を満たしてしまう．そこで，合計到着間隔 G が大きいほど A_{min} の値を大きく，逆に G が小さいほど A_{min} の値を小さくするよう調整を行うようにした．本研究では A_{min} を非攻撃時における 1s あたりの間隔個数と定義し，$N(c(h,t)) \geq G(c(h,t) \times A_{min}$ を満たすときに攻撃開始判定処理を行う．例えば，通常時において 1s あたり 40 個の間隔個数が観測されている場合を考える．この時，あるセル $c(h,t)$ の集約している期間が 3s だとすると，そのセルの持つ間隔個数 $N(c(h,t))$ が $120(=3\times40)$ 以上の場合に攻撃開始判定処理を行う．

提案手法の攻撃検知の流れを図 6，図 7 に示す．

5. 評価実験

本章では，以前に著者らが提案した手法 [15] に継続判定処理を追加したことにより長時間継続する同一レートの攻

表 1　実験環境

CPU	Intel(R) Core i7-4770 @ 3.40GHz
メモリ	8GB
OS	ホスト OS: Ubuntu 16.04.1 LTS, ゲスト OS: Ubuntu 12.04.5 LTS(メモリ 1GB 1 コア)
開発環境	C++

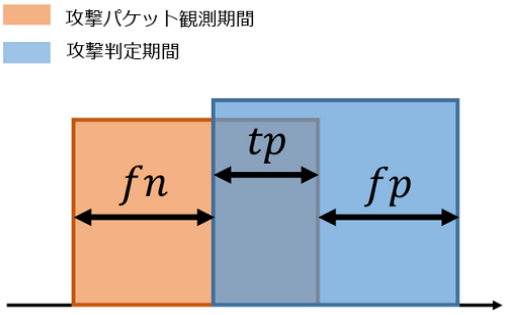

図 8　tp, fp, fn の計算方法

撃に対応できるようになったことを，5.3 節で検証する．次に，5.4，5.5 節で，検知精度と処理性能について既存の手法と比較することによって提案手法の有効性を確認する．具体的には，本研究と同じく即応性に着目した小島らの手法 [5] との比較対照を行う．以下では [5] を比較手法と呼ぶ．

5.1　実験環境

比較のため，比較手法と極力環境を同じにして評価実験を行った．全ての実験に共通する実験環境を表 1 に示す．Ubuntu 16.04.1 が稼働する研究室 PC 上に Ubuntu 12.04.5 が動作する仮想環境を構築しゲスト OS 上で実験を行った．ゲスト OS はメモリ 1GB，CPU1 コアで稼働している．なお，比較手法も同じく仮想環境上で実験を行っていたが，仮想ホストの CPU クロック周波数が明記されていなかったため，クロック周波数の調整は特に行わなかった．

提案手法のパラメータ値は，全ての実験において $n = 50, \beta = 0.01, W_{min} = 1.0$ に設定している．これらは経験的に定めた値である．

A_{min} の値については，通常時の特徴を抽出する王らの手法 [17] で提案された閾値学習アルゴリズムを活用して求めた．本研究では A_{min} を非攻撃時における 1s あたりの間隔個数と定めているが，この値を得るためには，ある組織における通常時の間隔個数を算出する必要がある．また，間隔個数が A_{min} 以上のときに攻撃の疑いがあるとして攻撃開始判定処理を行うようにするため，正常と思われる範囲での最大値を A_{min} として算出できる方法が望ましい．王らの手法では，学習データに依存することなく，ある任意の値についての正常範囲の最大値を抽出できるので，この手法によって A_{min} の値を計算することが適当だと考えた．

王らの手法はポートスキャンを検知するための手法である．まずポートアクセス数の度数分布を作成し，正常な領域と異常な領域とに分類した後，正常な領域における最大のポートアクセス数を攻撃検知の閾値として設定する．閾値の学習にパラメータ調整を必要としないためデータ依存の無い検知が可能である．

本研究では，学習用のキャプチャデータから間隔個数の度数分布を作成し，計算結果の値を A_{min} に利用した．度数分布は bin 幅を 10，時間単位を 1s に設定して作成した．なお，間隔個数が 0 の場合は A_{min} の決定に必要ないと判断し，分布に反映させなかった．

実験データおよび学習用データは 5.3 節と 5.4・5.5 節で異なるため，各節の「実験方法」の項で詳細を説明する．

5.2　評価式

本研究では，fp(false positive) と fn(false negative) を総合的に評価する F 尺度を検知精度の評価に利用する．攻撃検知システムにおいて，誤検知である fp と fn(false negative) は，一般的に fp を小さくすると fn が大きくなり，fn を小さくすると fp が大きくなる傾向にある．しかし，F 尺度はそれらを総合的に評価することができる．F 尺度は情報検索の分野で利用される指標であるが，文献 [18]，文献 [19] など DDoS 攻撃検知の分野でも利用されている．

F 尺度の計算式を式 (5) に示す．F 尺度は値が大きいほど良いとされ，値の範囲は $0 \leq F \leq 1$ である．

$$F = \frac{2PR}{P + R} \tag{5}$$

ここでの P と R はそれぞれ適合率（Precision）と再現率 (Recall) と呼ばれる値で式（6），式（7）によって定義される．

$$P = \frac{tp}{tp + fp} \tag{6}$$

$$R = \frac{tp}{tp + fn} \tag{7}$$

F 尺度は，提案手法によって出力された攻撃判定期間と実際の攻撃観測期間を比較し，重なっている時間を tp(true positive)，攻撃観測期間のうち判定期間が含まれない時間を fn，判定期間のうち攻撃観測期間が含まれない時間を fp として計算することで求まる（図 8）．なお，今回の実験では攻撃が行われていることを positive とする．

5.3　継続判定処理の追加による効果の検証

本節では，継続判定処理の追加によって長時間継続する同一レートの攻撃の終了判定を正確に行えるようになったことを検証する．

5.3.1　実験方法

実験では，検知対象となる長時間継続する同一レートの攻撃によるパケットを，ある期間だけ含むような送信パ

ケットデータ列を用意し，これを提案手法を組み込んだ仮想マシンに送ったときに，攻撃パケットの送られていた期間を提案手法が正しく検知できることを確認する．

実験データとして，非攻撃時のキャプチャデータと攻撃パケットの含まれるキャプチャデータを人工的にマージしたものを利用する．マージの方法として mergecap を利用した．Wireshark に付属されるツールの 1 つで，パケットデータをタイムスタンプ順にマージする．

非攻撃時のデータとして，研究室 PC のホスト OS 上で 10 分間ブラウジングを行ったときのキャプチャデータを利用した．続いて，攻撃時のデータとして，hping3 というパケットジェネレータで 10000pps のレートの UDP パケットを 3 分間送信したものを利用した．具体的には仮想環境上にキャプチャホストと攻撃ホストの 2 台のゲスト OS を用意し，攻撃ホストからのパケットをキャプチャした．

攻撃パケットは，最初に送信される攻撃パケットのタイムスタンプが，ちょうど非攻撃時のパケットキャプチャを開始した 5 分後に設定されるように送信を開始した．こうした理由は，現在の提案手法はデータ構造を十分に構築し終えてからでないと攻撃検知が行えないためである．攻撃検知を行うためには，最低でも n 個のレベル 0 セルが生成されなければならない．今回の実験では攻撃終了検知の精度を観測したいため，攻撃開始は確実に検知できるよう，データ構造を構築し終えてから攻撃が到着するように時間を調整した．上述のキャプチャデータを，仮想ホスト上で稼働する提案手法に入力として与え実験を行った．また A_{min} 決定のための学習用データとして，同じくホスト OS 上で 10 分間ブラウジングを行ったパケットキャプチャデータを利用した．学習の結果から $A_{min} = 150$ に設定した．

5.3.2 結果と考察

実験の結果，継続判定処理を組み込むことにより，長時間継続する同一レートの攻撃に対して途中で攻撃の検知が出来なくなる問題点を改善できたことが分かった．

実験データのパケット量の時間変化のグラフに，実際の攻撃期間と，攻撃を受けていると判定された期間を色付けしたものを，それぞれ図 9，図 10 に示す．グラフの縦軸はパケット数，横軸はキャプチャデータのうちパケットが最初に観測された時刻からの経過時間（秒）である．グラフのうち赤で示されている期間が実際に攻撃が観測されている期間で，青で示されている期間が提案手法によって攻撃だと判定した期間である．図 9 が継続判定処理が無い場合，図 10 が継続判定処理がある場合の結果である．図 9 を見ると，継続判定処理が無い場合は，パケット量が多いにも関わらず，途中で攻撃検知出来なくなっていることが分かる．一方図 10 を見ると，継続判定処理がある場合は，問題なく攻撃を検知できていることが分かる．検知精度を計算した結果が表 2 である．結果から分かるように検知精度が全体的に改善されており，長時間継続する同一レートの攻撃に対して提案手法が正しく検知できることを確認できた．

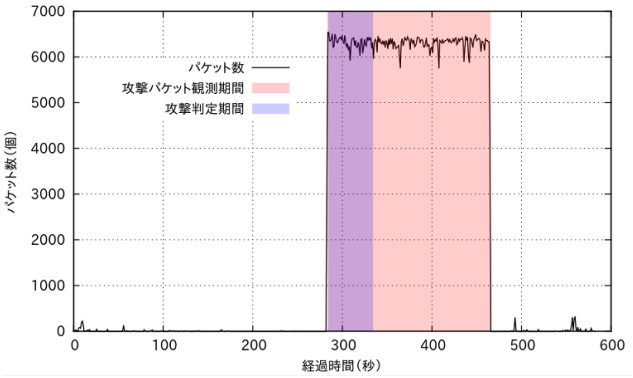

図 9　継続判定処理が無い場合の結果

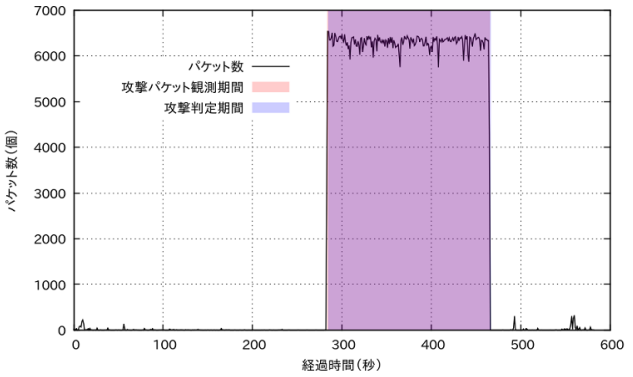

図 10　継続判定処理がある場合の結果

表 2　F 尺度の結果

	適合率	再現率	F 尺度
継続判定処理なし	1	0.274	0.431
継続判定処理あり	0.995	0.995	0.995

5.4 提案手法の検知精度

本節では提案手法の検知精度について，比較手法の検知精度と比較することで有効性を確認する．

5.4.1 実験方法

実験では，攻撃パケットの含まれたキャプチャデータを実験データとし，攻撃パケットが送られていた期間を正しく検知できることを確認する．実験データは 5.3 とは異なり，MIT Lincoln Laboratory が人為的に作成した DDoS 攻撃のデータセットである DARPA2000[20] を利用した．このデータセットは 2000 年のものであるため古いものとなっているが，DDoS 攻撃検知の研究では評価に広く利用されている．DARPA2000 は組織に侵入しホストをボット化した後に外部組織に DDoS 攻撃を仕掛けるまでのキャプチャデータが保存されており，DDoS 攻撃の他にも，IP スキャン，侵入などの攻撃パケットが観測されている．DARPA2000 は以下の 5 段階のシナリオを想定している．

Phase1　リモートホストからターゲット組織に対して IP

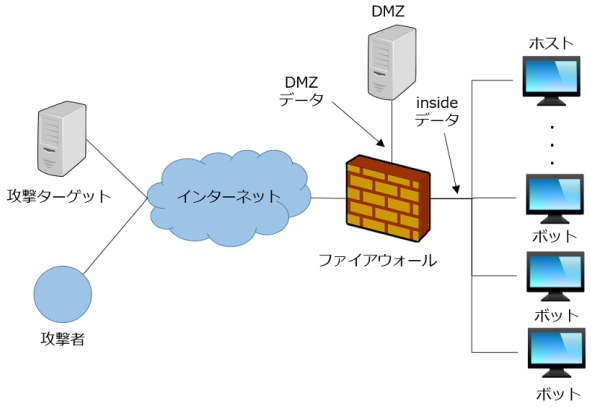

図 11　DARPA2000 の環境の模式図

スキャンを行い，稼働しているホストを調査

Phase2　Phase1 で調査したアクティブなホストに対し sadmind の動作の有無を確認

Phase3　sadmind の脆弱性を利用し，システムに侵入

Phase4　3 台のホストに DoS 攻撃ツールの mstream をインストールしボット化

Phase5　ボットに外部組織に対する DDoS 攻撃の開始を指示

DARPA2000 ではターゲット組織のファイアウォールの内部で観測された inside データと外部で観測された DMZ データが提供されている (図 11)．本研究では総パケット量の多い inside データ（総パケット数 649,787）を利用し，Phase5 の DDoS 攻撃部分のみを攻撃とみなす．

A_{min} 決定のための学習用データは，DARPA2000 と同程度の規模のバックグラウンドトラフィックが保存されている DARPA1999[20] を利用した．DARPA1999 は DARPA2000 と同じ組織によって作成されたキャプチャデータである．5 週間分のデータが用意されており，第 3 週までが訓練データとして，第 4 週，5 週が評価データとして提供されている．訓練データのうち，第 2 週には攻撃が含まれているが，第 1 週，第 3 週は攻撃が含まれていない．本研究では非攻撃時における間隔個数の決定のため，攻撃の含まれない DARPA1999 の第 1 週の火曜日のキャプチャデータを学習データとして用いた．火曜日のデータを利用しているのは，DARPA2000 が火曜日に観測されたデータであるためである．学習の結果から，$A_{min} = 490$ に設定する．

5.4.2　結果と考察

実験を行った結果，F 尺度については，提案手法は比較手法に比べて低い値となったが，一次検知としては十分な精度を持っていると考えられた．

5.3 節と同様に，パケット量の時間変化のグラフに，攻撃期間と攻撃判定期間を示す色付けをしたものが図 12 である．ここから F 尺度を求めた結果を表 3 に示す．提案手法の F 尺度は 0.906 となっており，検知精度が下がってしまっていることが分かる．しかしながら，fp を考慮した検知精度である適合率は 0.895，fn を考慮した検知精度である再現率は 0.917 と，どちらも 0.9 付近の結果となっている．今回，fn は攻撃パケットが最初に到着した時刻と攻撃開始検知時刻のずれを，fp は攻撃パケットが最後に到着した時刻と攻撃終了検知時刻のずれを表していることになる．攻撃を検知できなかった期間は攻撃初期の段階であり，fp の割合も 1 割程度であることから一次検知としては十分な精度を持っていると考えた．

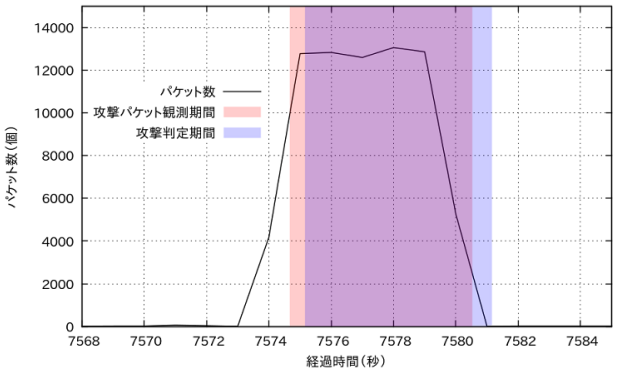

図 12　攻撃パケット観測期間と攻撃判定期間の結果

表 3　F 尺度の結果

適合率	再現率	F 尺度	比較手法 [5] の F 尺度
0.895	0.917	0.906	0.993

表 4　処理時間の結果

提案手法	比較手法
0.127s	61s

本研究ではパケットの到着情報のみを検知に利用しているため，他の統計情報を利用して有効な閾値を設定することで検知精度が向上すると考えられる．

5.5　提案手法の処理性能

本節では提案手法の処理性能について，比較手法の処理性能と比較することで有効性を評価する．処理性能は，DARPA2000 の inside データに含まれる 649,787 パケットの処理に要した時間で評価を行う．提案手法の値は実験を 10 回行った際の平均値となっている．実験方法，実験データ，および A_{min} の値は 5.4 節と同じである．

実験の結果，提案手法は比較手法に比べてはるかに早く処理を終えることができた．実験での処理時間を表 4 に示す．提案手法は比較手法よりも 60.873s 速く処理を終えている．表 4 の値は，二つの手法について異なる CPU クロック周波数の下で得られたものであるが，それを考慮しても処理性能は提案手法の方が高いといえる．

また，この結果から 1 パケットあたりの処理時間を計算すると 0.195μs となり，十分な即応性を持つといえる．提案手法では，イベント情報の圧縮処理による複数のパケッ

トの処理を，パケットが実際に到着してから行わなければならないが，それを考慮しても十分な性能である．

6. まとめ

本論では，リアルタイムバースト検出手法を利用したDDoS 攻撃の検知手法を提案した．パケットが到着するごとに攻撃検知処理を行うことによりリアルタイムな解析が可能となり，データ圧縮処理によって大量のパケットが到着しても効率的に対応できる．また，同一レートの攻撃が長時間継続すると攻撃が続いているにも関わらず攻撃終了と判断してしまう，過剰に攻撃判定処理を行うという問題の解決のため，新たに追加した攻撃継続判定，A_{min} の調整処理について説明した．

次に，以前の著者らの手法と提案手法の検知結果を比較し，長時間継続する同一レートの攻撃に対応できるようになったことを検証した．実験の結果，検知精度が改善されており，攻撃の終了判定を正確に行えるようになったことが分かった．

そして，検知精度と処理性能について比較手法と比較し提案手法の有効性を調査した．評価実験の結果，検知精度は比較手法と比較すると低いものの，一次検知としては十分な検知性能を持っていた．現在はパケット到着の情報のみを攻撃検知に利用しているため，他の統計情報を利用することで検知精度が向上すると考えられる．また，比較手法よりも処理性能が高く，1 パケット当たりの処理時間が 0.195μs となっていることから，圧縮処理による複数パケットの処理を，パケットの到着時に行うことを考慮しても，十分な即応性を持っているといえる．

さらに，比較手法では数万パケット程度の学習サンプルを必要としており学習過程の検討が必要であるが，提案手法は特別な学習過程は必要としない点において優れている．

今回 A_{min} 以外のパラメータ値は経験的に設定したものであるため，今後の課題として，パラメータ値の有効な値を自動的に獲得する仕組みが望まれる．特に β は通常時と攻撃時のパケット流入量によって大きく変動するので，検討が必要である．また，今回は全てのパケットを監視対象としているが，ある攻撃に特有なパケットを監視することで多様な攻撃に対応できると考えている．

参考文献

[1] L. Garber: Denial-of-Service Attacks Rip the Internet, Computer, pp. 12-17(2000).

[2] P. Vixie, G.Sneeringer, Mark Schleifer: “Events of 21Oct2002”(2002) available at:`http://c.root-servers.org/october21.txt`(accessed 2017/02/07).

[3] S. Arukonda,S. Sinha: The incorrect perpetrators: reflectors and reflection attaks., ACSIJ Advances in Computer Science: an International Journal, Vol. 4, Issue 1, No.13 pp.94-98(2015).

[4] 鈴木聖子:“史上最大級の DDoS 攻撃に使われたマルウェア「Mirai」公開、作者が IoT を悪用”,ITmedia エンタープライズ available at:`http://www.itmedia.co.jp/enterprise/articles/1610/04/news046.html`(2016) (2017/01/25 参照).

[5] 小島 俊輔, 中嶋 卓雄, 末吉 敏則: エントロピーベースのマハラノビス距離による高速な異常検知手法, 情報処理学会論文誌 Vol.52 No.2 pp.656-668(2011).

[6] O.Osanaiye,K-KR Choo, M. Dlodlo: Distoributed denial of Service(DDoS) resilence in cloud: Review and concenptual cloud DDoS mitigation framework, Journal of Network and Computer Applications 67, pp.147-165(2016).

[7] 蛯名 亮平, 中村 健二, 小柳 滋: リアルタイムバースト検出手法の提案, 日本データベース学会論文誌, Vol.9, No.2 pp.1-6(2010).

[8] Snort Homepage: available at:`https://www.snort.org`(2011)(accessed 2017/02/08).

[9] L. Feinstein, D. Schnackenberg, R. Balupari, D. Kindred: Statistical Approaches to DDoS Attack Detection and Response, Proceeding of DARPA Information Survivability Conference and Exposition, Vol.1, pp.303-314(2003).

[10] Y. Ohshita, S. Ata, M. Murata: Detecting Distributed Denial-of-Service Attacks by Analyzing TCP SYN Packets Statistically, IEICE Tran. Commum, vo.E89-B,No.10 pp.2868-2877(2006).

[11] K. Giotis, C. Argypoulos, G. Androulidakis, D. Kalogeras, V. Maglaris: Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments, Computer Networks 62,pp.122-136(2014).

[12] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, M. Rajarajan: A Survey of intrusion detection techniques in Cloud, Journal of Network and Computer Applications 36, pp.42-57(2013).

[13] X. Zhang, D. Shasha: Better Burst Detection,ICDE’06 Proceeding of the 22nd International Conference on Data Engineering, pp.146-149(2006).

[14] 蛯名 亮平, 中村 健二, 小柳 滋: リアルタイムバースト解析手法の提案, 情報処理学会論文誌 データベース,Vol.5, No.3 pp.86-96 (2012).

[15] 臼崎 翔太郎, 橘 弘智, 有川 佑樹, 高塚 佳代子, 山場 久昭, 久保田 真一郎, 岡崎 直宣: リアルタイムバースト検出手法を利用したパケット到着間隔による DDoS 攻撃検知手法の検討, 電気・情報関係学会, 第 69 回電気・情報関係学会九州支部連合大会 p.270(2016).

[16] 原田 薫明, 川原 亮一, 森 達哉, 上山 憲昭, 廣川 裕, 山本 公洋: 異常トラヒック発生検出および終了判定手法電子情報通信関係学会, 信学技報, IN2006-133 pp.115-120(2006).

[17] 王 サン, フォン ヤオカイ, 川本 淳平, 堀 良彰, 櫻井 幸一: 挙動に基づくポートスキャン検知の自動化に向けた学習アルゴリズムの提案とその性能評価, 情報処理学会論文誌 Vol.56 No.9 pp.1770-1781(2015).

[18] Y. Gu, A. McCallum, D. Towsley: Detecting Anomalies in Network Traffic using Maximum Entropy Estimation, Proc. Internet Measurement Conference. Berkeley, CA,US,pp.345-350(2005).

[19] 小島 俊輔, 中島 卓雄, 末吉 敏則: 複合攻撃への検知精度を向上させる χ^2 手法の提案と評価, 情報処理学会論文誌,Vol.53,No2. pp.836-846(2012).

[20] MIT : DARPA Intrusion Detection Evaluation Data Set. availarble at:`https://www.ll.mit.edu/ideval/data/index.html`.