Regular Paper

A Proposal of Highly Responsive Distributed Denial-of-Service Attacks Detection Using Real-Time Burst Detection Method

Shotaro Usuzaki[1] Yuki Arikawa[1] Hisaaki Yamaba[2] Kentaro Aburada[2,a)]
Shin-Ichiro Kubota[3] Mirang Park[4] Naonobu Okazaki[2]

Received: June 9, 2017, Accepted: December 8, 2017

Abstract: Distributed Denial-of-Service (DDoS) attack detection systems are classified into a signature based approach and an anomaly based approach. However, such methods tend to suffer from low responsiveness. On the other hand, real-time burst detection which is used in data mining offers two advantages over traditional statistical methods. First, it can be used for real-time detection when an event is occurring, and second, it can work with less processing as information about events are compressed, even if a large number of events occur. Here, the authors add the function for attack detection in real-time burst detection technique, and propose a highly responsive DDoS attack detection technique. This paper performs experiments to evaluate its effectiveness, and discusses its detection accuracy and processing performance.

Keywords: Distributed Denial-of-Service attack, real-time burst detection, highly responsive

1. Introduction

Distributed Denial-of-Service (DDoS) attacks are a type of cyber attacks which disables the target server from providing the service for users by sending illegal traffic to the server. DDoS attacks are a major threat to society because attacks have been reported against many vital services such as major portal sites [1] and root DNS servers [2]. In addition, the attack of about 421 Gbps in 2014 [3] and about 620 Gbps from IoT equipment in 2016 [4] have also been reported, therefore, the damage is getting bigger every year. For these reasons, it is necessary to develop an effective DDoS attack detection system.

The anomaly based approach detects DDoS attacks by comparing the statistics of a packet sequence with precomputed values under normal conditions. The sequence length of packets used for calculation is called window size, and is measured either in time or packet count. The anomaly based approach has the advantage of being able to reduce the false negative rate for unknown attacks. However, the responsiveness decreases with increasing window size, because the anomaly method cannot detect attacks if the packet counts or time do not exceed a predefined value [5]. For example, one entropy-based approach [6] needs a window size of 10,000 packets.

Since the anomaly-based method cannot detect the start and the end of an attack until the window size is exceeded, in the case of failure to detect the attack-start quickly, it cannot quickly move to attack mitigation processing by detecting packets which are suspected to be from attackers, which causes the damage of the attack to spread.

Next, we present the need for attack-end detection. In general firewalls, all packets suspected of being attacks are dropped; therefore, if the detection process is continued for a long period of time even though the attack is over, service cannot be provided to the legitimate users because their packets are dropped. Considering that the network-based services are now fundamental infrastructure, if the services such as banking ATM and other financial and medical services become unavailable, social confusion can occur. Consequently, it is necessary to finish the attack mitigation process as soon as possible.

Therefore, we propose a method to detect the start and the end of an attack at each packet arrival by using the burst detection algorithm while maintaining a wide window size. This technique is already used in data mining to analyze data streams, and is applied in this paper to analyze the burst for each event occurrence to detect attacks more quickly than analyzing the packets at predetermined window size. The target of the proposed method is focused on detection of a DDoS attack, and it is assumed that the subsequent attack mitigation processing is performed by another method. The goal in this method is to detect the start and the end of an attack quickly and to support the immediate shift to the implementation and cancellation of subsequent regulation.

[1] Graduate School of Computer Science and Systems Engineering, University of Miyazaki, Miyazaki 889–2192, Japan
[2] Faculty of Engineering, University of Miyazaki, Miyazaki 889–2192, Japan
[3] Center for Management of Information Technologies, Kumamoto University, Kumamoto 860–8555, Japan
[4] Faculty of Information Technology, Kanagawa Institute of Technology, Atsugi, Kanagawa 243–0292, Japan
a) aburada@cs.miyazaki-u.ac.jp

This paper is structured as follows: Section 2 describes the related study of DDoS attack detection method. Section 3 describes the real-time burst detection technique and the data structure. Then, we discuss its benefits for DDoS attack detection. Section 4 presents the details of the mechanism and formula for anomaly detection. Section 5 shows the results of performance evaluation experiments using the proposed method. Section 6 discusses the conclusions and future work.

2. Related Study

Generally, the DDoS attack detection systems are classified into two categories: signature based and anomaly based approach. However it has been pointed out that both approaches have a problem because of their low responsiveness.

Signature based detection technique monitors a packet pattern and compares the features and knowledge database. Snort [7] is a typical method of signature based approach. A signature of Snort includes the source/destination IP address, port number, payload, and options such as metadata. When a packet arrives, an attack is detected by comparing the signature with a pre-registered one. Signature based approach has high detection accuracy of existing attacks if the signature is always updated to the latest [8]. Also, this approach easily has the advantage of the addition and management of a signature [9]. On the other hand, this approach cannot detect the subspecies or unknown attacks. Furthermore, this approach may take a time to detect the attack when many signatures are registered [5].

Reference [6] is a type of entropy technique that uses entropy values as statistical information. The IP address, port number, etc. included in the header field of the packet are used as random variables, and the entropy value is calculated for each fixed window size. Attack detection is performed by utilizing the fact that the amount of entropy such as the destination IP address and the destination port number decrease when packets concentrate at the DDoS attack. An advantage of the entropy method is that it is difficult for an attacker to make an attack that does not exceed the threshold value because the attacker cannot know the entropy value of the target organization. In addition, calculation of the entropy value is mainly for counting; therefore, the speed of computation is higher than that of the pattern matching methods. However, with the entropy method, it is necessary to widen the window size to improve detection accuracy. This is because, if the window size is narrow, it causes a large fluctuation of the entropy, and it becomes difficult to detect by the threshold value. Reference [6] said that a window size needs more than 10,000 packets in order to lower the false detection rate, as it is not necessary for responsiveness to be high.

Reference [10] is a detection method of SYN Flood attack. The SYN packet arrival rate at normal times follows the normal distribution and it disappears at an abnormal time. By utilizing the fact that the SYN packet arrival rate at a normal time follows a normal distribution and does not follow at an abnormal time, packets are sampled at regular time intervals to calculate the mean square error, and detection is carried out when it deviates from the normal distribution. In this method, because the arrival of packets transmitted intentionally is detected, effective detection can be performed even when the attack packet amount is small. However, there is overhead in calculating the mean square error using the normal distribution, and high-speed computability is lacking.

Reference [11] is a DDoS attack mitigation method in the Software Defined Network (SDN) environment. The system consists of three modules: packet collection, attack detection, and attack mitigation, and the entropy method is applied to the attack detection module. This method can detect not only DDoS attacks but also worms and port scans by monitoring a plurality of changes in the entropy value of each information source. In the packet collection module, the unit of the window size is time, and it is set to 30 seconds. Since detection processing is started after collecting packets, there is a problem that detection can not be performed at least until the determined window size is exceeded. In addition, it takes at least 30 seconds for the end of the attack to be known; therefore, there is a possibility that legitimate packets which arrived during that time may be falsely judged as part of a suspected attack.

3. Real-Time Burst Detection Method

The burst detection method is used in data mining to analyze abnormalities in a data stream. A data stream is a high-speed data flow such as that generated by online news, blog, and internet forums. An abnormal aggregate in the data stream is called a burst. Detecting the burst quickly allows spotlighted information to be immediately extracted from a massive data flow. The method proposed by Ebina et al. [12], hereinafter referred to as the real-time burst detection method, can detect bursts more quickly than methods that analyze data streams at regular intervals, since each individual event is processed. In addition, it reduces the number of unnecessary calculations when events do not occur, because the data structure is updated only when an event occurs.

Moreover, by compressing the information for events that arrived in a window $W_{\min}$, which is the minimum window size, a large number of events can be processed efficiently. The reason for introducing $W_{\min}$ is that when analyzing the burst for each event, the processing load is increased when a large number of events occur in a short period time.

3.1 Data Structure

The data structure used in this method is based on the aggregation pyramid (AP) proposed by Zhang and Shasha [13]. As shown in **Fig. 1**, the AP is composed of a plurality of cells, and is an n-level pyramid-shaped data structure built within a time window of size n. The level h cell that ends at time t, denoted as $c(h, t)$, stores the total time gap between the first and last event within

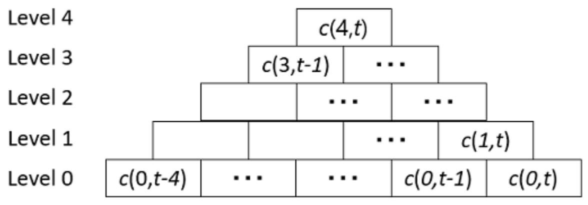

Fig. 1 Aggregation pyramid for $n = 5$.

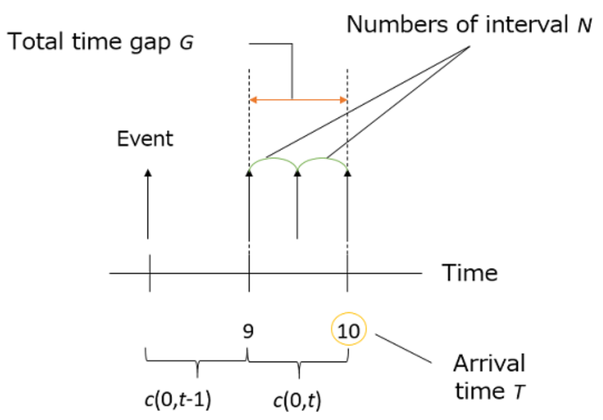

Fig. 2 Information contained in cells.

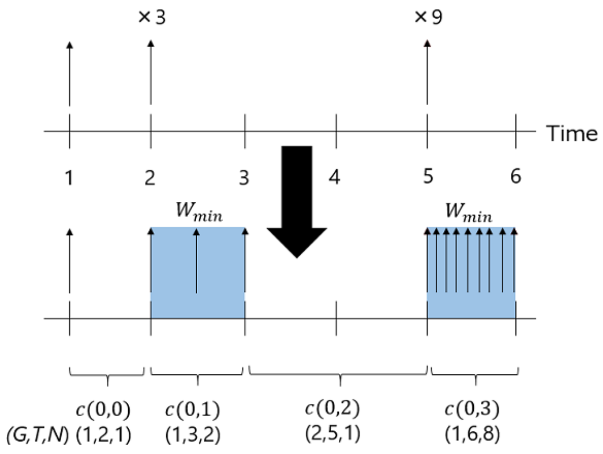

Fig. 3 Procedure for creating level 0 cells.

a cell $G(c(h,t))$, arrival time $T(c(h,t))$, and number of intervals $N(c(h,t))$ (**Fig. 2**).

3.2 Cell Generation

This subsection describes the construction of a data structure in the real-time burst detection method. The data structure is obtained by generating cells with the procedure described in the following rule and shown in **Fig. 3**.

In the rule, we define the time intervals when a series of $m + 1$ events occur by $\boldsymbol{x} = (x_1, x_2, \ldots, x_m)$, we will use this notation in the following explanation of the rule. Actually, it is not necessary to wait for $m + 1$ packets to arrive because x_i is calculated at the arrival of every packet.

If $x_i < W_{\text{min}}$ holds, this method performs a compression process. The cell is compressed to length W_{min} by setting $T(c(0,t))$ equal to $T(c(0,t-1))+W_{\text{min}}$. Events corresponding to (arrival time of packet) $< T(c(0,t-1)) + W_{\text{min}}$ are compressed and the number of these events is calculated in order to determine $N(c(0,t))$, which is the number of intervals. If no event occurred just at $T(c(0,t))$, then the number of events occurring is decremented by 1 under the general procedure for getting the number of intervals (upper part of **Fig. 4**). Then, the values are set as $N(c(0,t))$. Through this process, it is assumed that the last event that satisfies (arrival time of packet) $< T(c(0,t-1)) + W_{\text{min}}$ occurred at $T(c(0,t))$, and it is placed at the end of the cell, and corrected

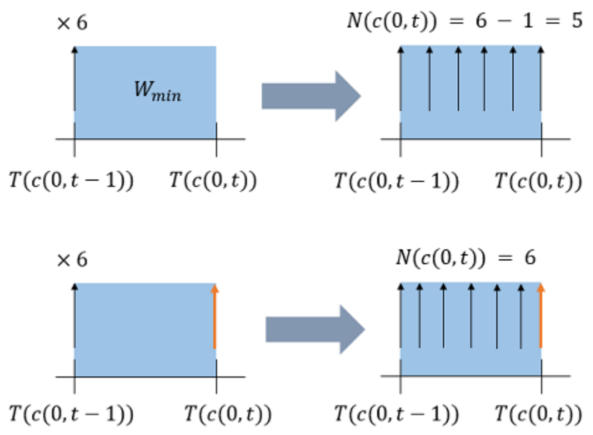

Fig. 4 Details of calculation method for the $N(c(0,t))$ in compression process.

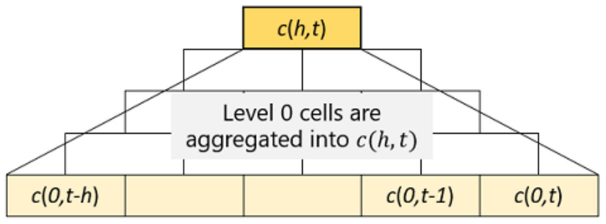

Fig. 5 Information held by $c(h,t)$.

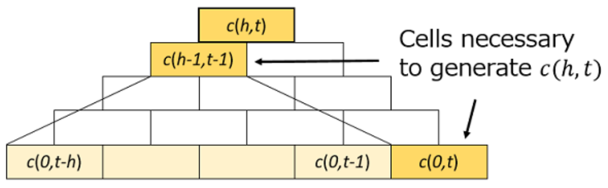

Fig. 6 The cells required by $c(h,t)$.

so that the events occurred evenly during W_{min}. However, if the next event occurs at exactly $T(c(0,t))$, that event becomes the end of the cell (the orange arrow in Fig. 4). In this case, the event count number is set as $N(c(0,t))$ because the number of intervals increases by one (lower part of Fig. 4).

A level 0 cell $c(0,t)$ has the first-hand information of event arrival. In addition, A cell $c(h,t)$ has arrival informations from $c(0,t-h)$ to $c(0,t)$ (**Fig. 5**), by being calculated from the informations of $c(h-1,t-1)$ and $c(0,t)$ (**Fig. 6**).

(1) How to generate level 0 cell.

(a) $x_i \geq W_{\text{min}}$
- $G(c(0,t)) = x_i$
- $T(c(0,t))$ = the event arrival time of the (i+1)th event.
- $N(c(0,t)) = 1$
- $i = i + 1$

(b) $x_i < W_{\text{min}}$
- $T(c(0,t)) = T(c(0,t-1)) + W_{\text{min}}$
- $N(c(0,t))$ = the number of event occurrences from $T(c(0,t-1))$ to $T(c(0,t))$
- if no event occurred at time $T(c(0,t))$, $N(c(0,t)) = N(c(0,t)) - 1$
- $G(c(0,t)) = W_{\text{min}}$
- $i = i + N(c(0,t))$

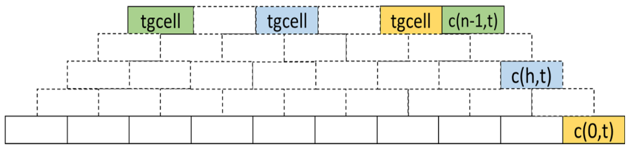

Fig. 7 Cells used for comparison during burst detection.

- x_i = elapsed time from $T(c(0,t))$ to the occurrence of the next event.
- if multiple events occur at $T(c(0,t))$, $x_i = 0$

(2) How to generate level h cell.

- $G(c(h,t)) = G(c(h-1,t-1)) + G(c(0,t))$
- $T(c(h,t)) = T(c(0,t))$
- $N(c(h,t)) = N(c(h-1,t-1)) + N(c(0,t))$
- if the cell $c(h,t)$ belongs to max level$(h = n-1)$, $t = t+1$

As described above, the number of redundant data updates can be reduced compared to techniques with a regular update process, because the data structure is updated only when an event occurs.

3.3 Burst Detection Methodology

In burst detection methods, when a cell is generated, the mean event arrival interval for a cell and the nearest top level $(n-1)$ cell, denoted *tgcell*, that does not overlap the aggregated window is compared. In **Fig. 7**, these pairs of cells are shown in the same color.

The average event arrival interval is defined as the average value of the arrival intervals that are aggregated within a cell. This value is calculated by Eq. (1) using $G(c(h,t))$ and $N(c(h,t))$. Furthermore, the burst strength for the cell is defined by Eq. (2).

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

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

Then, a parameter β is set for determining the occurrence of a burst in Eq. (3). Since the above detection process is performed as soon as an event occurs, this method provides real-time burst detection.

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

Next, a parameter $A_{\min}$ is introduced in order to suppress excessive burst detection. This is because a burst will be detected when the arrival interval undergoes a large change even if the number of packets is small. Thus, an event is defined as a burst only if $N(c(h,t)) \geq A_{\min}$ is satisfied.

4. Proposed Method

Our study used the real-time burst detection method to detect DDoS attacks. We assume that the DDoS attack considered is at least larger than a DoS attack, and detection is carried out under the assumption that the DDoS attack can be captured by using arrival interval information which does not need to consider address differences. However, this method has the drawback of not being able to detect attacks that occur at a constant rate for a sustained period, and frequently erroneously detects attacks.

Here, we describe our approach to overcome these problems.

4.1 Addition of Continuous Attack Detection Methodology

Although the detection method using Eq. (3) is suitable for sensing the beginning of a short attack, if the attacks continue for a long time at the same rate, it is no longer able to detect them. This is because it is difficult to determine the difference in the packet average arrival interval for the current and previous cell. A situation where a large number of packets arrive can not be detected despite attacks will cause enormous damage, so measures are required.

To detect a continuous attack occurring at a constant intensity, we also monitor the spread of the differences in the arrival intervals. Attacks occurring at a constant rate are assumed to result in a reduction in the differences in the arrival intervals. By monitoring whether the arrival interval is widened or not, it is expected that a sustained attack occurring at a constant rate can be detected.

In the proposed method, the system is always in one of two different states: attack-start detection or attack-end detection. When the start of an attack is detected using Eq. (3), the objective of the detection method changes from detecting the start of an attack to detecting the end of an attack. Our method determines whether an attack has ended or not using the following equation.

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

If the above equation does not hold, the attack is deemed to have ended and the state returns to attack-start detection after outputting $T(c(0,t-1))$ as the attack end time. The attack detection period is defined as the time between the start and end of the attack.

To determine whether an attack is continuing, the reciprocal of the burst strength $brt(begin)$ at the start of the attack is set as the threshold value. The reason why the threshold value is determined as described above is because the packet count during normal periods is considered to be almost constant. If it is judged as an attack due to the interval being $1/x$ times the previous one, we assume that the attack has ended when it returns to the interval before attack because the interval is widened x times. For example, if the average packet arrival interval becomes $1/10$ times shorter than the previous one, we detect the attack-end when the spread increases by 10 times.

In the attack-end detection state, the current cell $c(0,t)$ is compared with the average packet arrival interval using the adjacent level 0 cell $c(0,t-1)$, but not the *tgcell*, because we assume that detection of continuity does not require distant past information. Moreover if the attack is judged to have been terminated in the cell following the one in which it was first detected, the attack is regarded as not having occurred. This is because it is considered to be due to the arrival of several packets simultaneously.

In addition, we assume that attack-end detection will lead to quick detection of the end of the attack. In the network management process for abnormal traffic, the importance of quickly detecting attack termination is shown in Ref. [14]. Its purpose is to promptly cancel underway regulations so as not to affect legitimate users. Since the proposed method detects attack-end by Eq. (4), we assume that it can detect an attack quickly in case of a large difference in packet flow between attack and normal time.

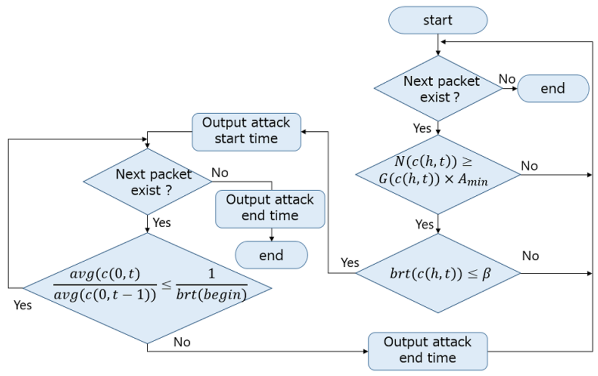

Fig. 8 Algorithm for proposed attack detection method.

4.2 Suppression of Excessive Detection Events

In the previous method, $A_{\min}$ was set to a constant value. However, When a cell $c(h, t)$ belongs to highly level, the number of intervals $N(c(h, t))$ increases because of the aggregation of many level 0 cells, and $N(c(h, t)) \geq A_{\min}$ will hold even if an attack is not occurring.

Therefore, we increase/decrease $A_{\min}$ as the total interval $G(c(h, t))$ increases/decreases. This study defines $A_{\min}$ as the number of intervals per second when an attack is not occurring. The beginning of the attack detection process occurs when $N(c(h, t)) \geq G(c(h, t)) \times A_{\min}$.

The flow of the proposed method is shown in **Fig. 8**.

5. Evaluation Experiment

We evaluated the effectiveness of the proposed method by comparing its detection accuracy and processing performance with those for an existing method. Specifically, we compared our proposed method with that proposed by Oshima et al. [5], whose focus is on high responsiveness. In the following, this method will be referred to as the comparative method.

5.1 Computing Environment

For ease of comparison with the original study, we have tried to keep our computing environment as similar as possible to that used by Oshima et al. [5]. The computing environment used in all experiments is shown in **Table 1**.

We ran the experiments on a virtual machine using Ubuntu 12.04.5 as the guest operating system (OS) on a laboratory PC with Ubuntu 16.04.1 as the host OS.

Parameter settings of $n = 50, W_{\min} = 1.0, \beta = 0.01$ were used in all experiments. These values were determined empirically.

The $A_{\min}$ threshold was obtained using the learning algorithm proposed in [15] for extracting features at normal conditions.

In this study, $A_{\min}$ is defined as the number of intervals per second at non-attack time. In order to obtain this value, it is necessary to calculate the number of intervals at the normal time for each organization. Additionally, it is desirable to set the maximum value of the normal range to the value of $A_{\min}$. Because it is determined that there is a suspected attack, and the attack-start detection processing is performed if the number of intervals N is more than $A_{\min}$.

With this algorithm, it is possible to extract the maximum range of values considered normal for a certain arbitrary value without depending on the learning data, making it suitable for calculating the value of $A_{\min}$. This technique is used to detect when a port scan is occurring. In this method, the frequency distribution of the port access count is classified into normal and abnormal regions, and then the maximum number of port accesses in the normal region is set as the threshold for attack detection.

After we fitted a frequency distribution for the interval number from the capture data for learning, we calculated $A_{\min}$ using the above method. The distribution was fitted with time steps of 1 second and a bin width of 10 seconds. If the interval number was 0, we did not use it in the distribution fitting because it was considered unnecessary for determining $A_{\min}$.

Experimental and learning data are different in Section 5.3 and 5.4–5.5, so we will explain in detail in the "Experiment Method" of each section.

Table 1 Evaluation environment.

CPU	Intel(R) Core i7-4770 @ 3.40 GHz
Memory	8 GB
Host OS	Ubuntu 16.04.1 LTS
Guest OS	Ubuntu 12.04.5 LTS (Memory: 1 GB CPU: 1 Core)
Programming Language	C++

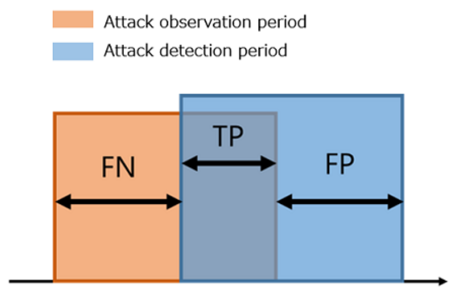

Fig. 9 Method for calculating tp, fp, and fn.

5.2 Evaluation Index

We used the F-measure to objectively evaluate the fp (false positive) and fn(false negative) rate in the evaluation of the detection accuracy.

In attack detection systems, fn generally tends to increase if fp is decreased, and vice versa. However, the F-measure can be used to evaluate the accuracy of a model. Although the F-measure is used in the field of information retrieval, it is also used to evaluate the accuracy of DDoS attack detection as in Refs. [16] and [17]. The formula for calculating the F-measure is given in Eq. (5). The F-measure takes values of $0 \leq F \leq 1$, and the larger the value, the better the detection accuracy. We also considered the precision and recall values for the method, which are defined by Eqs. (6) and (7), respectively. The F-measure is calculated from tp(true positive), fp and fn, which are obtained as shown in **Fig. 9**. P and R stand for precision and recall, respectively, where P represents the ratio of attacks that were really attacks in the attack detection period and R is the ratio detected as an attack during the attack observation period.

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

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

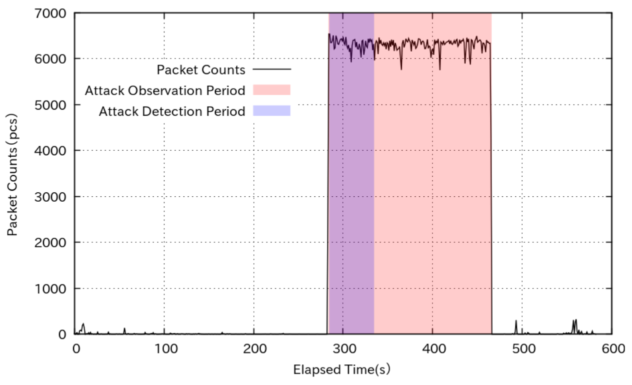

Fig. 10 Observed and estimated attack periods in case of non attack-end detection.

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

5.3 Verification of Effect by Adding Attack-End Detection

In this experiment, we confirm that the method of Ref. [12] cannot detect the attack-end due to prolonging the detection period, and show that the problem is solved by the improvement of our proposal.

5.3.1 Experiment Method

We prepare packet data containing packets of target attacks for only a certain period, and confirm that the proposed method can correctly detect the period during which the attack packet is being transmitted.

As experiment data, we use artificially merged capture data including non-attack and attack packet. We used a merging tool called mergecap, it is a tool to merge the packet file in time stamp order which it is attached to Wireshark. As non-attack data, we used capture data of browsing for 10 minutes on the host OS of lab laboratory PC. For the attack data, we used a packet generator called hping3, and sent a UDP packets with a rate of 10,000 pps for 3 minutes. The attack packets were transmitted so that the time stamp of the first packet was set 5 minutes after the time stamp of the first non-attack packet data.

The reason for preparing the above data is that attack detection can not be performed unless the data structure is fully constructed. In order to perform attack detection, at least n level 0 cells must be generated. In this experiment, we had to observe the accuracy of attack end detection, so we adjusted the time stamp so that the attack would arrive after constructing a sufficient data structure.

Also, as learning data for decision of $A_{\min}$, we used other data that was browsing for 10 minutes on the host OS.

5.3.2 Results

As a result of this experiment it was found that a constant rate attack can be detected by incorporating the improvements during the process. **Figures 10** and **11** show the attack observation and the detection period in the graph of the packet count. The period during which attacks are actually observed is indicated in red, and the period during which attacks are determined by the proposed method is indicated in blue.

Figure 10 shows the result of the case where there is no continuation determination processing, and Fig. 11 shows the result of the opposite case.

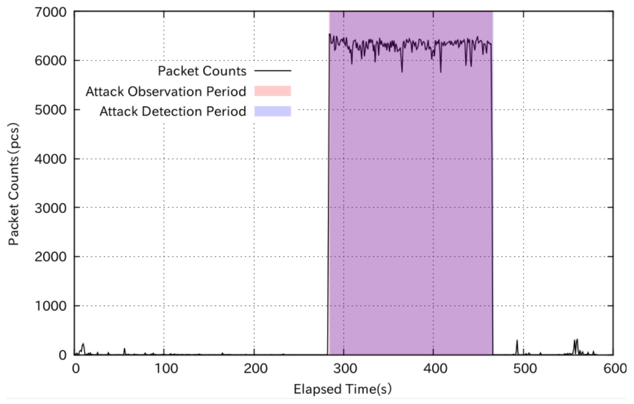

Fig. 11 Observed and estimated attack periods in case of attack-end detection.

Table 2 Estimated F-measures.

	Precision	Recall	F-measure
Non Attack-End Detection	1	0.274	0.431
Attack-End Detection	0.995	0.995	0.995

As can be seen from Fig. 10, when there is no continuation determination processing, an attack cannot be detected even though the packet amount is large. On the other hand, as shown in Fig. 11, by adding the attack-end detection processing, it can be seen that attacks can be detected without any problem. **Table 2** shows the result of calculating the detection accuracy. As can be seen from the results, the detection accuracy is improved and the proposed method can correctly detect for the constant rate attack for a long time.

Because this experiment aims to clarify that there is vulnerability in the method [12], we create a constant long-lasting attack that causes the vulnerability in a simple way. It is necessary to evaluate with another data set in the future because this data set is not a DDoS attack but a DoS attack. In addition, generic communication is not included in the used data set, and attacks are also simple; therefore, it is not possible to evaluate whether the accuracy of attack-end detection improved in this experiment. In order to evaluate the accuracy of detection of the termination of a DDoS attack, there is a need for using a data set that is closer to the actual environment.

5.4 Evaluation of Detection Accuracy

In this section, the effectiveness of the proposed method is evaluated by comparing its detection accuracy with that of the comparative method.

5.4.1 Experiment Method

In the experiment, capture data including attack packets with a known attack duration were used as input for the proposed method. We used the DARPA 2000 [18] synthetic data set from the MIT Lincoln Laboratory as the experimental data.

The DARPA 2000 attack scenario includes five phases as shown below.

Phase 1 Perform an IP scan from the remote host to the target organization and investigate the host that is running.

Phase 2 Confirm whether sadmind works on active host investigated in Phase 1.

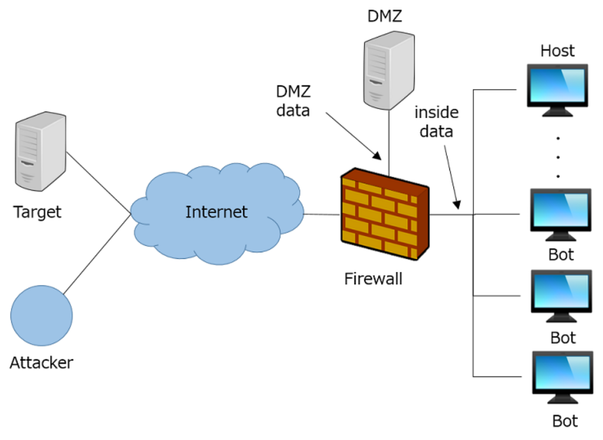

Fig. 12 Schematic drawing of network in DARPA 2000.

Phase 3 Break into the system by using sadmind's vulnerability.

Phase 4 Install the DDoS attack tool mstream on three hosts and make them a bot.

Phase 5 Instruct the bot to attack against the target organization.

The data observed inside the firewall of the target organization and the DMZ data observed outside the firewall are provided in **Fig. 12**. We used the inside data with the largest total packet count, and considered the attack packets in Phase 5 as the objects to be detected. For estimating $A_{\min}$, we also used DARPA 1999 [18], which has the same amount of background traffic as DARPA 2000.

In DARPA 2000, packets for DDoS attacks are transmitted from internal organizations to external attack targets; therefore, this data set cannot be exactly modeled in experiments as traffic arriving at the attack target. In this experiment, we evaluate DARPA 2000 as traffic data arriving at the external organization based on the assumption that the same amount of traffic flows as the ordinary traffic of DARPA 2000's inside data in the attack target organization because this method does not use packet header information and so on.

DARPA 1999 contains seven weeks of training data. For the learning data, we used Tuesday in the first week of data, which did not contain any attacks. In addition, the DARPA 2000 data was also recorded on a Tuesday. Here, $A_{\min}$ was set to 490 from the results of the learning stage.

5.4.2 Results

From the results of the experiment, the proposed method had a lower value for the F-measure than the comparative method. However, we consider that the proposed method has sufficient accuracy for primary detection.

Figure 13 shows attack observation and detection in the graph of the packet count. The reason why the result shows only a part of the graph is because the attack period is about 5 seconds, which is short with respect to the entire time of experimental data. Additionally, **Table 3** shows the estimated F-measures.

The F-measure for the proposed method was 0.906, and the precision and recall were around 0.9. In this experiment, fn represents the difference between the arrival time of the first attack packet and the attack start time, as detected by the proposed method. Similarly, fp is the difference between the arrival time of the final attack packet and the estimated attack end time. Since attacks could only go undetected in their early stages, and the value of fp was 10%, we consider our method to have sufficient accuracy for primary detection.

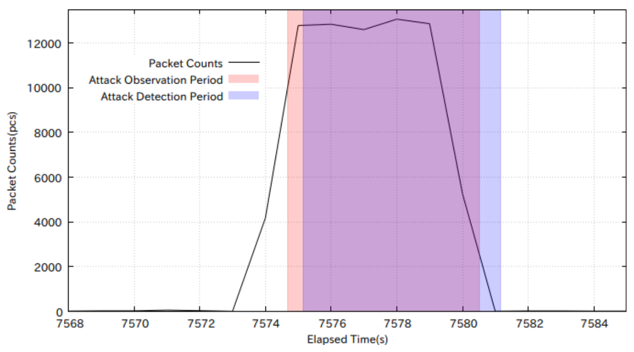

Fig. 13 Observed and estimated attack periods.

Table 3 Estimated F-measures.

Precision	Recall	F-measure	F-measure for comparative method [5]
0.895	0.917	0.906	0.993

Table 4 Processing time.

Proposed Method	Comparative Method
0.127 s	61 s

5.5 Evaluation of Computational Efficiency

In this section, we evaluate the computational efficiency of the proposed method by comparing it with that of the comparative method. The computational efficiency is evaluated by the time taken to process the 649,787 packets contained in the DARPA 2000 inside data. The estimated time for the proposed method was averaged over 10 experiments. We used the same experiment and learning data as in 5.4.

The results of the experiment show that the proposed method finishes 60.873 faster than the comparative method (**Table 4**).

The computational efficiency of the proposed method is high, although it should be noted that the values in Table 4 were obtained with different CPU clock speeds for the two approaches. Further, the processing time per packet was estimated to be 0.195 microseconds. The proposed method has to process multiple packets by compressing the packet arrival information after packets arrive. However, even with this data compression, the proposed method has sufficient computational efficiency to be used in DDoS detection.

5.6 Evaluation of Responsiveness

In Ref. [5], responsiveness is evaluated from the relationship between the window size [packets] and the F-measure. A high responsiveness is defined as when the window size is small and the F-measure is high. The experimental results in Ref. [5] shows that their method can detect an attack even if $W = 10$. The minimum window size is defined as 1 second in the proposed method. In addition, about 19.5 packets on average arrive per second for the inside data of DARPA 2000. According to this result, there is a possibility that it has the same degree of responsiveness although it is inferior to the comparative method. However, the comparative method requires learning samples of about tens of thousands of packets and it is necessary to study the learning process. The proposed method is superior to that method because our method can detect attacks when at least $2 \times N$ packets arrive. Furthermore, the value fluctuation is considered to be small because the proposed method can hold the window size to at least $n \times W_{\min}$. We need to investigate the detection accuracy when $W_{\min}$ is further reduced.

5.7 Evaluation of Followability

In Ref. [5], followability is evaluated by experiments in which four patterns of traffic are added to the inside data of DARPA 2000. Followability means that an attack can be detected even if the traffic volume varies depending on the time zone and is evaluated by calculating the F-measure in the four patterns. The four patterns are (1) the tendency of the destination IP address is in the concentration state from the distribution state, (2) it is in the distribution state from the concentration state, (3) the tendency of the source IP address is in the concentration state from the distribution state, and (4) it is in the distribution state from the concentration state. This additional traffic has a packet amount corresponding to 50% of the inside data of DARPA 2000. The detection process uses a plurality of the information sources, but the response of these entropy values is similar to that of the destination and the source IP addresses. Therefore, these four patterns are regarded as simulating all common traffic. This method has a high following capability from the result which a F-measure of 0.952 or more at $W \geq 20$ excluding one pattern. The packet interval information used in the proposed method exists in various patterns.

It is difficult to have high detection accuracy for each pattern as the followability of the proposed method is decreased in comparison with the comparative method. In order to increase followability of the proposed method, it is necessary to change effective parameter values, especially β, in real time.

6. Conclusion

In this paper, we proposed a burst detection method for the rapid detection of DDoS attacks. Our method enables real-time analysis, as it can be reapplied as each packet arrives, and can efficiently process information from a large number of attack packets with its use of a data compression process.

We introduced an extension of our methodology for detecting attacks with a continuous arrival rate, and for suppressing excessive detection.

We investigated the accuracy and efficiency of the proposed method by comparing it with a comparative method. Although the detection accuracy was lower than for the comparative method, our method had sufficient accuracy to be used for primary detection of an attack. In particular, there is a possibility of false detection when communicating to exchange a large-capacity file because the proposed method detects attacks by monitoring a decreasing packet arrival interval. As the countermeasure, it is conceivable to use capture data in which large-scale communication is frequently observed as $A_{\min}$'s learning data. In the case of an organization where video transfer is frequently carried out, the algorithm [15] for extracting the feature at the normal time outputs the value considering the situation. By this countermeasure, it can be prevented from entering the attack-start detection judging that the burst-type flow is not suspected of being an attack. Furthermore, there is a risk of false detection if large-scale communication is performed within the same LAN as the server to be monitored because this method does not distinguish packets using the source IP address. In order to solve this problem, it is necessary to revise so as not to monitor packets whose source and destination IP addresses are private addresses.

Also, in the attack-end detection state, if the attacker adjusts the attack to gradually reduce the attack interval, the attack detection period becomes significantly long without entering the attack-start detection state, resulting in a denial of service condition. As a countermeasure, it is conceivable to introduce a judgment formula by $A_{\min}$ and to make the transition to the attack start state when the number of intervals falls below $A_{\min}$. However, if the attacker knows the value of $A_{\min}$, this countermeasure will fail; therefore, further investigation is necessary.

We used only packet arrival information for detection, and the accuracy of our method could be further improved by utilizing other statistics. In addition, we believe that it can be used to detect a variety of types of attack by monitoring the specific packets of a given attack. The efficiency of the proposed method was higher than that for the comparative method.

All of the data set used in this experiment were artificially created. In the data set, although ordinary TCP communication such as access to a portal site or getting an image file is being performed, it does not include the communication of requesting a large-capacity file, such as a video which is expected to be included in actual traffic. The occurrence of a large amount of burst-like data transfer may lead us to increase false positives. On the other hand, attacks in each data set have steadily arrived by being performed as attacks from within the same LAN in addition to the small number of attack hosts. In actual DDoS attacks, the number of hosts is large, and when the hosts belonging to botnets are in different organizations, the arrival interval of attack packets may be sparsely scattered. In that case, there is a problem that attack-start detection is delayed or the attack-end detection finishes early. In order to confirm the above problem, it is necessary to use a data set close to the actual traffic for the experiment. In particular, DARPA 2000 does not reproduce the actual DDoS attack packet amount because these data have only 3 attack hosts. Also the amount of data in ordinary communication is small compared to a real environment because these data do not include large-capacity file and encryption exchanges. In this method, the detection precision and speed are affected by packet differences, since these results may decrease due to a change in effective β value. Furthermore, in the case of operating in a real environment, there is a possibility that a longer processing time is required than in the experiment because more packets arrive than the data set. Moreover, this data set does not include burst flows such as uploading/downloading large-capacity files, which causes more false positives, which were not observed in the experiment.

In the present study, parameter values were determined empirically. However, it is necessary to be able to automatically obtain valid values for the parameters. This is especially the case for β, since this value will fluctuate due to differences between packets under normal and attack conditions. Along with that, it is necessary to investigate conditions for distinguishing between normal traffic and DDoS attack packets after completion of detection because the current method does not distinguish between attacks and normal packets.

Acknowledgments This work was supported by JSPS KAKENHI Grant Numbers JP17H01736, JP17K00139.

References

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

[2] Vixie, P., Sneeringer, G. and Schleifer, M.: Events of 21Oct2002 (2002), available from ⟨http://c.root-servers.org/october21.txt⟩ (accessed 2017-02-07).

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

[4] Krebs, B.: 21 KrebsOnSecurity Hit With Record DDoS (2016), available from ⟨https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/⟩ (accessed 2017-06-06).

[5] Oshima, S., Nakashima, T. and Sueyoshi, T.: Fast Anomaly Detection Method Using Entropy-based Mahalanobis Distance, *IPSJ Journal*, Vol.52, No.2, pp.656–668 (2011), in Japanese.

[6] Feinstein, L. and Balupari, R.: Statistical Approaches to DDoS Attack Detection and Response, *Proc. DARPA Information Survivability Conference and Exposition*, pp.303–314 (2003).

[7] Snort Homepage, available from ⟨https://www.snort.org⟩ (accessed 2017-02-08).

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

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

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

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

[12] Ebina, R., Nakamura, K. and Oyanagi, S.: A Real-Time Burst Detection Method, *2011 23rd IEEE International Conference on Tools with Artificial Intelligence* (*ICTAI*), pp.1040–1046 (2011).

[13] Zhang, X. and Shasha, D.: Better Burst Detection, *Proc. 22nd International Conference on Data Engineering*, *ICDE'06*, pp.146–149 (2006).

[14] Harada, S., Kawahara, R., Mori, T., Kamiyama, N., Hirokawa, Y. and Yamamotom, K.: A method of detecting network anomalies and determining their termination, *IEICE Technical Report, Information networks*, Vol.106, No.420, pp.115–120 (2006). in Japanese.

[15] Wang, C., Feng, Y., Kawamoto, J., Hori, Y. and Sakurai, K.: A Learning Algorithm for Behavior-based PortScan Automatic Detection and Its Evaluation, *IPSJ Journal*, Vol.56, No.9, pp.1770–1781 (2015). in Japanese.

[16] Gu, Y., McCallum, A. and Towsley, D.: Detecting Anomalies in Network Traffic using Maximum Entropy Estimation, *Proc. Internet Measurement Conference*, pp.345–350 (2005).

[17] Behal, S. and Kumara, K.: Detection of DDoS attacks and flash events using novel information theory metrics, *Computer Networks*, Vol.116, pp.96–110 (2017).

[18] MIT: DARPA Intrusion Detection Evaluation Data Set, available from ⟨https://www.ll.mit.edu/ideval/data/index.html⟩.

Shotaro Usuzaki was born in Miyazaki, Japan, in 1994. He received his B.S. degree from the Faculty of Engineering, University of Miyazaki, Japan, in 2017. He is studying Computer Science with the Graduate School of Engineering, University of Miyazaki. He is a student member of IPSJ.

Yuki Arikawa was born in Kagoshima, Japan, in 1994. He received his B.S. degree from the Faculty of Engineering, University of Miyazaki, Japan, in 2016. He is studying Computer Science with the Graduate School of Engineering, University of Miyazaki.

Hisaaki Yamaba received his B.S. and M.S. degrees in chemical engineering from the Tokyo Institute of Technology, Japan, in 1988 and 1990, respectively, and the Ph.D. degree in Systems Engineering from University of Miyazaki, Japan, in 2011. He is currently an Assistant Professor with the Faculty of Engineering, University of Miyazaki, Japan. His research interests include network security and user authentication. He is a member of IPSJ, SICE and SCEJ.

Kentaro Aburada received his B.S., M.S and Ph.D. degrees in Computer Science and System Engineering from University of Miyazaki, Japan, in 2003, 2005 and 2009, respectively. He is currently an Associate Professor with the Faculty of Engineering, University of Miyazaki, Japan. His research interests include computer network and security. He is a member of IPSJ and IEICE.

Shin-Ichiro Kubota received his Ph.D. degree from Kumamoto University in 2006. He has been an Associate Professor at Kumamoto University since 2017. He has also worked at Kagoshima University (2003–2007), Kumamoto University (2007–2013) and University of Miyazaki (2013–2017). His research interests contain online-learning, e-portfolio and web technologies. He is a member of IPSJ, ACM and IEEE.

Mirang Park received her B.S. and M.S. degrees in Electrical Engineering from Hanyang University, South Korea, in 1983 and 1985, respectively, and Ph.D. degree in Information Science and Technology from Tohoku University, Japan, in 1993. She is currently a Professor with the Faculty of Information Technology, Kanagawa Institute of Technology, Japan. She joined the Information Technology Research and Development Center, Mitsubishi Electric Corporation, in 1994, where she has been involved in the research and development of network security. Her research interests include security for large deployment of sensor networks, peer-to-peer networks, and user authentication. She is a member of IPSJ, IEICE, JSSM and IEEE.

Naonobu Okazaki received his B.S., M.S and Ph.D. degrees in Electrical and Communication Engineering from Tohoku University, Japan, in 1986, 1988 and 1992, respectively. He joined the Information Technology Research and Development Center, Mitsubishi Electric Corporation, in 1991. He is currently a Professor with the Faculty of Engineering, University of Miyazaki since 2002. His research interests include mobile network and network security. He is a member of IPSJ, IEICE and IEEE.