# DDoS Attack Detection Algorithms Based on Entropy Computing

Liying Li[1], Jianying Zhou[2], and Ning Xiao[3]

[1] National University of Singapore, Singapore
liliying@alumni.nus.edu.sg
[2] Institute for Infocomm Research, Singapore
jyzhou@i2r.a-star.edu.sg
[3] Symantec Software Dev. (Chengdu) Co. Ltd, China
ning_xiao@symantec.com

**Abstract.** Distributed Denial of Service (DDoS) attack poses a severe threat to the Internet. It is difficult to find the exact signature of attacking. Moreover, it is hard to distinguish the difference of an unusual high volume of traffic which is caused by the attack or occurs when a huge number of users occasionally access the target machine at the same time. The entropy detection method is an effective method to detect the DDoS attack. It is mainly used to calculate the distribution randomness of some attributes in the network packets' headers. In this paper, we focus on the detection technology of DDoS attack. We improve the previous entropy detection algorithm, and propose two enhanced detection methods based on cumulative entropy and time, respectively. Experiment results show that these methods could lead to more accurate and effective DDoS detection.

**Keywords:** DDoS detection, entropy computing, network security.

## 1 Introduction

The traditional *Denial of Service* (DoS) attack is usually a point-to-point attack. The attacker makes use of proper service requests to occupy excessive service resources to force the server crash, or to make other legal users unable to attain timely service responses. When the host under attack has limited computing, memory and network bandwidth, the consequence of DoS attacks could be fairly serious. However, along the development of computer and network technology, the impact of DoS attacks has been significantly mitigated.

*Distributed Denial of Service.* (DDoS) attack is an extension of the traditional DoS attack. DDoS attack is a kind of distributed, cooperative large-scale attack. It has the same working principles as DoS, but compared with the traditional DoS whose attack is originated from a single attacker point, the realization of DDoS comes from hundreds or even thousands of PC attackers which have been installed Daemon, and it is a group-based attack behavior. The targets of DDoS are usually high-volume websites, search engines, or government departments.

Compared with the traditional DoS attack, DDoS attacks possess more attacking resources and have more destroying power, and thus they are more difficult to be detected and defended. DDoS attacks have brought tremendous threat to the security of Internet, and also gain much research attention in the area of network security [4, 20].

Now, the DDoS attacks tend to become more distributed and automated, and the destruction is more serious. The attacks have some technical trends: (1) make use of clusters of controlled PCs to start intensive attacks; (2) produce randomly distributed source IP addresses to conceal the track; (3) change the structure of attack packets randomly; (4) explore the bugs and weaknesses of both network protocols and operating systems; (5) send packets faster with no apparent attack characteristics. Hybrid attacks make the defense even harder.

Once the DDoS attacks have been carried forward, the attack packets will flood to the targeted victim and submerge those legal users' packets, making those legal users unable to access the server's resources. Only by timely detection of DDoS attacks, the system could make proper response to escape big loss. Research conducted by other organizations shows that statistical measurements and processing is an effective approach to DDoS problem [12]. The EMERALD project at SRI International uses intrusion detection signatures with Bayesian inference to detect distributed attacks [14]. A destination address monitoring scheme was proposed in [17]. Using only a few observation points, the authors proposed a method to monitor the macroscopic effect of DDoS flooding attacks [7]. In [2], the authors detect flooding attacks at the edge and classify them as incoming or outgoing attacks with an *Artificial Neural Network* (ANN).

In this paper, we put forward two new DDoS detection methods based on the traditional entropy detection method [1, 8]. One uses computing cumulative entropy, which monitors the network for a period of time instead of making judgment soon after detecting abnormal network condition. The other method makes use of the concept of time to judge the network condition without setting a threshold of traffic volume, but observing whether the abnormal network condition persistently lasts for a certain period. We conduct experiments for the traditional entropy detection and the cumulative entropy detection methods, respectively. The test results demonstrate that our improved methods have better detection capability than before.

The rest of this paper is organized as follows. We briefly introduce the background of DDoS attack detection in Section 2, then propose two new approaches based on cumulative entropy and time, respectively in Section 3. Section 4 describes our implementation, and Section 5 shows the experiment results. Finally, we conclude the paper in Section 6.

## 2 Previous Work

In this section, we first introduce the background of DDoS attack detection, and then focus on the entropy detection algorithms which would be the premise of our improved algorithms shown in Section 3.

### 2.1 DDoS Detection Background

In the past years, it was discovered that DDoS attack methods and tools are becoming more sophisticated, effective, and also more difficult to trace to the real attackers. On the defense side, current technologies are still unable to withstand large-scale attacks [3].

Defending the DDoS attacks involves three phases: before the attack, during the attack and after the attack. The first one is precaution, which needs a process or long time to deploy the network to guard against the attack. The last one is the second line of defense. Therefore, a practical way to defend the DDoS attack is to prevent the attack flow reach the target and to ensure its availability. Protection using history-based IP filtering is a method when facing the attack [18]. But the premise of defense is to detect the attack timely and exactly.

The main DDoS detection methods comprise two categories: *signature-based detection* and *anomaly detection*. Our research is focused on the anomaly detection.

**Signature-Based Detection.** Suppose that the intruder's activity could be expressed by a pattern that gives an accurate description of some known attack or intrusion manners. The purpose of this method is to detect whether the object's activity matches these patterns or not. This method could detect the known intrusions, but could do nothing for the new intrusions. The difficulty is how to derive the pattern that could present the phenomenon of intrusion and will not cover other normal behaviors at the same time.

This method has high accuracy for the attack detection, but it is not useful for those intrusions or attacks without experience knowledge. The updating of detection rules is always slower than the emergence of new attacks. At present, after a new bug is published on the Internet, we might find the attack method and codes for this bug next day, but the relative detection rules will come out after several days. The time gap between the new attack and the update of user's rules will give the intruders enough time to launch attacks. In addition, many published attack detection rules still have high error rate, and more and more hackers tend to not publicize the bugs they have found. Therefore, it is difficult to summarize the characteristics of those attacks. In [9], the authors propose to discover the DDoS attacking signature by analyzing the TCP/IP packet header against the well defined rules and conditions, and distinguish the difference between normal and abnormal traffic. A general feature space modeling methodology was presented to identify DDoS attacks. It changes the non-separable attacks to separable cases, and it also allows the unknown attacks potentially being identified by their own features [15].

**Anomaly Detection.** This method pre-defines the normal and abnormal system activities, and thus to identify the abnormal behaviors among all normal system activities. When the anomaly occurs, it should be detected and responded by alerting or prevention. Some anomaly detection system could allow users to define a threshold or baseline for a normal behavior. This baseline could be constructed by sample statistics, or neural network. Then the detection system works. When finding the behavior exceeds this baseline, the system gives an alarm. More specifically, it compares the detection record of the network communication condition with the normal record. When the difference is large, we say some anomaly occurs and the detection system will warn the intrusion in time.

In [13], the authors use energy function variance based on wavelet analysis to detect DDoS attack traffic. A covariance model was proposed to effectively differentiate between normal and attack traffic, and to some extents verifies the effectiveness of multivariate correlation analysis for DDoS detection [16].

The objects used in the anomaly detection include: attack flow speed, packet size and port distribution, distribution of the packet arrival time, concurrent traffic flow number, advanced protocol characteristics, in/out data rate, and so on.

## 2.2 Entropy Detection

In information theory, the Shannon entropy or information entropy is a measure of the uncertainty associated with a random variable. It can be interpreted as the average shortest message length, in bits, that can be sent to communicate the true value of the random variable to a recipient. This represents a fundamental mathematical limit on the best possible lossless data compression of any communication: the shortest average number of bits that can be sent to communicate one message out of all the possibilities is the Shannon entropy. This concept was introduced by Claude E. Shannon in his 1948 paper *"A Mathematical Theory of Communication"*.

This entropy detection method is mainly used to calculate the distribution randomness of some attributes in the network packets' headers. These attributes could be the packet's source IP address, TTL value, or some other values indicating the packet's properties. For example, the detector captures 1000 continuous data packets at a peak point, and calculates the frequency of each distinct IP address among these 1000 packets. By further calculation of this distribution, we could measure the randomness of these packets [8].

After analyzing the phenomenon of DDoS attack, we could know that, when the attack comes out, there will be large number of data packets, high volume of traffic flow, and many incomplete connection requests. The attackers always fabricate a lot of data packets, and the IP addresses of these packets are generally different and randomly distributed. The analysis of these characteristics could help us to detect the DDoS attack better.

Entropy could be calculated by computing a series of continuous packets. The entropy value gives a description about the corresponding random distribution of these source IP addresses. The bigger the entropy, more random the source IP is. The smaller the entropy, the narrower the distribution range of packets' source addresses is, and some addresses have quite high emergence probability. Under normal network condition, the entropy of network packets always fluctuates to some extent. But when the attacks come out, the entropy value will have perceptible changes. We could detect the change of the source IP distribution through monitoring the entropy change, and thus provide reasons for keeping or discarding those packets.

Next, we discuss the detection methods by analyzing the distribution of packet's source IP. This is also the entropy computing model used as the basis for our improved detection algorithms. The formula of entropy calculation is as follows [1]:

$$H = -\sum_{i=1}^{n} p_i \log_2 p_i \tag{1}$$

where $p_i$ is the emergence probability of each distinct source IP address, $n$ is the total number of packets being analyzed, and $H$ is the entropy.

In [8], the authors proposed an improvement of this entropy detection computing. In the implementation of their algorithm, the authors use a fixed-size sliding window to simplify the computation complexity of the entropy. The window size is $W$, the probability $p_i$ here equals to emergence probability of each distinct source IP address, that is the counts of one address divided by the total packet number. Therefore, we do not need to calculate all the packets' entropy value for our detection, but just compute $W$ packets' entropy value for our judgments.

A proficient attacker usually tries to defeat the detection algorithm by secretly producing flooding attack and simulating the monitors' expected normal data flow. After knowing some packet attributes' entropy values, these attackers could use the attack tools to produce some flooding with adjustable entropy values. By guess, test and summary, these attackers could probably know the normal entropy range in the monitors, and adjust their own flooding to match it, although it is not easy to realize.

## 3 Improved Entropy Detection Methods

In this section, we propose two improved DDoS detection methods based on entropy computing. One uses computing cumulative entropy, and the other is time-based.

### 3.1 Cumulative Entropy Detection

*Cumulative Sum* (CUSUM) is an algorithm from statistical process control that could detect the mean variation of a statistical process. CUSUM is based on the fact that if there is some change happened, the probability distribution of the random sequence will also be changed [10, 11].

Here we further improve the previous entropy detection algorithm by incorporating the idea of cumulative sum and variation detection [10, 11] to our own entropy approach and try to cumulate the entropy according to some rules, thus it will have more accurate DDoS attack detection rate.

In our DDoS attack method, suppose $X_n$ is the entropy value calculated by using sliding window[8] at each time interval of $\Delta_n$, and the random sequence $\{X_n\}$ is extracted as network service random model. In the normal occasion, this sequence is independent and distributed. Assume the variation parameter is the average value of sequence $\{X_n\}$. Before change, this value $E(X_n) = \alpha$ is very small, and always positive. Before attack, when the network is normal, the distribution of IP addresses should be stable, and have little randomness, thus the entropy value should be small. But when DDoS attack happens, this average value will increase suddenly, $E(X_n)$ will become far bigger than $\alpha$, and becomes a constant or dynamic value.

CUSUM algorithm[10, 11] has an assumption that in the normal case, the average value of the random sequence should be negative, and it becomes positive after change. Therefore, without losing any statistics properties, we transfer the sequence $\{X_n\}$ into another random sequence $\{Z_n\}$ with negative average value. Let $Z_n = X_n - \beta$, where $\alpha = \alpha - \beta$. In a given network environment, the parameter $\beta$ is a constant, used for producing a negative random sequence $\{Z_n\}$, and thus the entire negative value of $\{Z_n\}$ will not be cumulated along the time. In our detection algorithm, we define that $\beta = 2\alpha$. Assume that when the network entropy value becomes two times as the normal network entropy, we say that the network is abnormal, and then we start to cumulate. When the attack happens, $Z_n$ will suddenly become very large and positive. The detection threshold is the limit for the positive, which is the cumulative value of $Z_n$.

We use this recursive formula for cumulative sum:

$$\begin{cases} y_n = (y_{n-1} + Z_n)^+ \\ y_0 = 0 \end{cases} \tag{2}$$

where $x^+ = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}$, $y_n$ and represents the cumulative positive value of $Z_n$. The bigger $y_n$ is, the stronger the attack is. For the variation in time period $\tau_N$ (when $y_{\tau_n} \geq N$), the judgment function could be:

$$d_N(y_n) = \begin{cases} 0, & (y_n \leq N) \\ 1, & (y_n > N) \end{cases} \tag{3}$$

where $d_N(y_n)$ is the judgment at time $n$, the value 1 shows that attack happens, while 0 shows the normal case. $N$ is the detection threshold.

The advantage of this improved algorithm lies in that it comprises implicitly a concept of process cumulating. In the previous entropy detection algorithms, we always judge the network condition according to a threshold. For example, when the network entropy is bigger than a value, we say the network is abnormal or some attack happens. But this judgment may not be suitable in some occasions. For example, the traffic flow in the network suddenly increases, but the flow is actually from some legal users. The function of cumulating process is to avoid the false alarm when the network has something abnormal just at a time point. We need to cumulate the total entropy during a time period, and judge this value whether exceeds the limit or not, and the results in this way should be more accurate. From the equation $y_n = (y_{n-1} + Z_n)^+$, we know when $Z_n$ is fluctuating among negative and positive values, the cumulative value $y_n$ might finally be 0, or just very small positive value. In this case, the network may only suddenly become abnormal, or not stable, but it is not attacked, and our detection will not give false alarm.

In the non-parameter CUSUM algorithm [10, 11], the idea of sequential variation is first proposed. But its approach is to analyze the ratio between the new arrival IP number in a time unit and the total IP number, and thus construct a random sequence. To implement that algorithm, we need to create a database containing a large amount of legal IP address, and each time we should compare and calculate the number of all new IP in each time unit. The calculation is complicated and has low efficiency. In our improvement, we use the entropy statistics based on sliding window [8]. Because the nature of entropy, it could clearly show the distribution of source IP's randomness. By controlling the sliding window size, we could enhance the detection accuracy. For example, when the host has large traffic flow in normal work, and the IP is very distributed, we could properly increase the window size to have better detection.

The implementation and test results of this cumulative entropy detection algorithm will be shown in Sections 4 and 5.

## 3.2 Time-Based Entropy Detection

In the anomaly detection, we usually have to set a threshold value. When the statistics exceed this threshold, we say that the system is facing attacks. In the previous entropy detection algorithms, when a single value is beyond the range or a cumulative value exceeds a value, the system will give an alarm. The setting of this threshold is usually through experience, to some extent. For some systems' design, they could also get a proper threshold value by using neural network to study the normal network. Here, we do not consider neural network, but try to use some simple method to complete timely and accurate attack detection.

Based on the cumulative entropy detection described before, we make some improvement. Here, we give up the threshold value $N$ and do not cumulate the entropy. Instead, we propose a time-based entropy detection method. The main concept is to use time to judge the network condition, not according to a threshold value to judge the attack condition.

We calculate the network entropy $V$ using a fixed rate and time unit $t$. Suppose $X_n$ is the entropy value computed by sliding window in each time interval $\triangle_n$ . By the formula $Z_n = X_n - \beta$, set $\beta = 2\alpha$ (here $\beta$ could be other values according to the environment), so we get a random sequence $\{Z_n\}$ with negative average value. (The computation of $Z_n$ is the same as described in cumulative entropy method.) Let $V_j = Z_n$. Construct a vector $X$, with initial value $X_0^T = [-1 - 1 \cdots - 1 - 1]$. Vector $X$ has $n$ elements, and the initial value for each element is -1.

$$x_j^{new} = \Phi(v_j) = \begin{cases} +1, & v_j > 0 \\ -1, & v_j < 0 \quad (1 <= j <= n) \\ x_j^{old}, & v_j = 0 \end{cases} \tag{4}$$

Using the above update rule, we could update each $X_j$ according to the relative $V_j$. When the vector $X^T = [+1 + 1 \cdots + 1 + 1]$, or say all the elements of $X$

becomes +1, it shows that there is some attack in the network, and the alarm is triggered. The update of each $X_j$ is cyclic, and the value of $j$ is from 1 to $n$, then 1 to $n$ again. The advantage of this algorithm is that we could control the total attack detection time by setting those two parameters: $t$ and $n$, where $t$ is the data collection interval, and $n$ is the element number of vector $X$. For example, when $t = 2s$, $n = 15$, and the system will give an alarm only when the network abnormal entropy persists over 30s. A sudden traffic increase in a short time might still be a normal traffic, and we allow it. But if the network's anomaly lasts for more than 30s, or even longer, we could believe that, the system might be abnormal, and some attacks might happen.

The threshold-based approach is widely applicable, and it may lead to a more real-time and timely attack detection. But for the time-based approach, we tend to emphasize the time tolerance. In some allowable range, we could ignore the network anomalies. But only when exceeding our tolerable limit, defined by a time period, we regard the network is abnormal or attack happens. These two approaches may have their own advantages under different environments. In some cases, the DDoS detection that combines threshold-based and time-based approaches may be more efficient, and have fewer false alarms.

## 4 Implementation

In this system, we need to start two threads for handling. The first thread (statisticThread) is mainly responsible for capturing packets and buffering them. The second thread (analysisThread) is used to analyze the packets' properties, and is controlled by a timer.

### 4.1 statisticThread Analysis

Our statistic thread is designed mainly based on the modification of Winpcap [6]. Winpcap is a system for capturing and analyzing packets under the platform

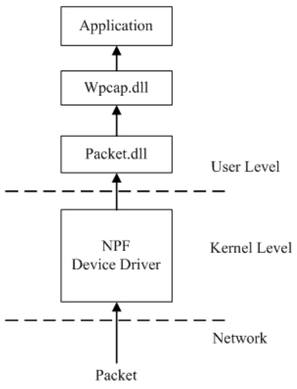

**Fig. 1.** Winpcap Processing

of Win32. It includes a kernel-level packet filter, a basic DLL (packet.dll) and an advanced DLL (Wpcap.dll) independent of the system as shown in Figure 1.

Based on the sliding window mode, we could buffer those packet contents we are interested in, and ensure the space utilization and computation convenience at the same time.

```
statistic_Info statisticWin[WINDOW_SIZE_RECORD];
```

Before the system runs, we need to create the database containing all legal IP address according to the system's history record. In order to hasten the system's running, we design a hash mechanism shown in Figure 2. First build a hash lookup table according to the IP address database, and realize the fast query for IP.

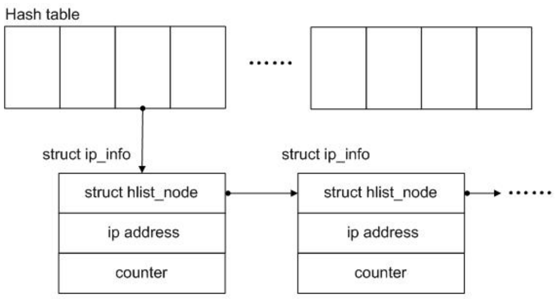

**Fig. 2.** Hash Table Structure

After invoking the function *pcap_next_ex*() to get the original packets from Winpcap, we enter the processing function. According to the current winPos, we calculate the saving address, and then save the information, and modify winPos.

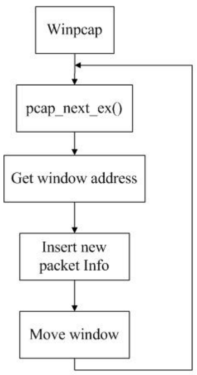

**Fig. 3.** statisticThread Processing

```
statistic_Info * pShannon = & statisticWin [winPos];
memcpy((u_char*)&pShannon->header, &pkt_data[26], sizeof(UINT32)*3);
pShannon->header.protocol = pkt_data[23];
winPos = (winPos + 1) % WINDOW_SIZE_RECORD;
```

The statisticThread Processing is shown in Figure 3.

### 4.2 analysisThread Analysis

The main function of the thread analysisThread is to save the data characteristics in the window. Set a timer that starts every 1s. Calculate the average value of the sequence $\{X_n\}$, and according to $Z_n = X_n - \beta$ and $y_n = (y_{n-1} + Z_n)^+$ to calculate the relative data. Plot the fluctuation graph, and judge whether the attack exists.

In the practical implementation, we start the detection analysis on those packets having the same IP address. The analysis thread starts every 1s, and we analyze the relative packets' characteristics during this 1s period, and conduct cumulating. When the cumulative value reaches a certain limit, the system will give an alarm.

## 5 Experiment Results

In this section, we show the experiment results and analyze the traditional entropy detection method and our two improved methods: cumulative entropy detection method and time-based detection method. In this test, our sliding window size is set to 1000, and the total test time is 300s.

### 5.1 General Entropy Statistics

Test method: we start two attacks [5]. The first one is between the time 40s and 80s, and the second one is from 180s to 230s. The effect of this entropy detection method is quite good, and the entropy value changes quickly, and increases to around 10. Here we only calculate every 1000 packets' entropy value. When the attack comes, the number of packets increases a lot, and a large number of random IP addresses come out. The extreme condition is that for every 1000 packets, each packet has a different IP address. In this case, the extreme and maximum value of entropy should be:

$$H = -\sum_{i=1}^{1000} p_i \log_2 p_i = -1000 \times 0.001 \times \log_2 0.001 \approx 9.966 \tag{5}$$

The test result is shown in Figure 4 below.

This experiment result is shown in the condition of one PC attacks another PC, and the CPU utilization of the attacked computer is reaching to 100% immediately. Note the configuration of the attacked computer has one CPU of Intel core 2 duo T5600, and its memory capacity is 512M. When we use two attackers to simulate the test, the attacked PC is directly shut down. Therefore, we could see the powerful destruction of TCP-SYN-Flood.

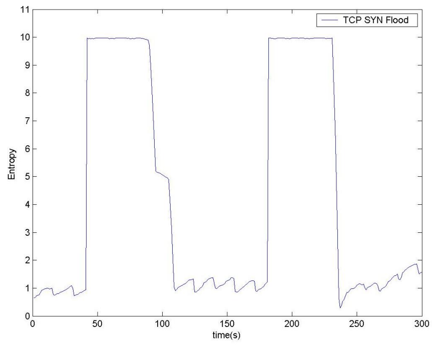

**Fig. 4.** Entropy Statistics under TCP-SYN-Flood

## 5.2 Cumulative Entropy Detection

We calculate the current packets' entropy every 1 second. From Figure 5, we could see that the normal entropy value is fluctuating around 2 bits in our network environment.

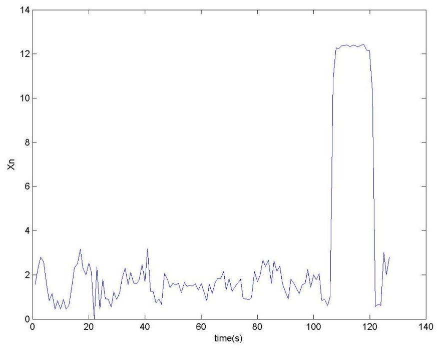

**Fig. 5.** Entropy Statistics $X_n$ in Network

We set $\beta = 2\alpha = 4$ bits, and then $Z_n = X_n - \beta$. In the normal condition, the sequence of $\{Z_n\}$ should be negative, sometimes $Z_n$ may be bigger than 0. But when the attack happens, $Z_n$ will rapidly increase a lot. As shown in Figure 6, the entropy becomes 10 bits, which is much bigger than 0.

By the formulas $y_n = (y_{n-1} + Z_n)^+$, $y_0 = 0$, we could cumulate the positive value of $Z_n$. As shown in Figure 7, in the normal case, $y_n$ should be 0 or a small positive value close to 0. When the attack happens, this cumulative value $y_n$ increases quickly. By setting the threshold $N$, when $y_n > N$, the system will give an alarm.

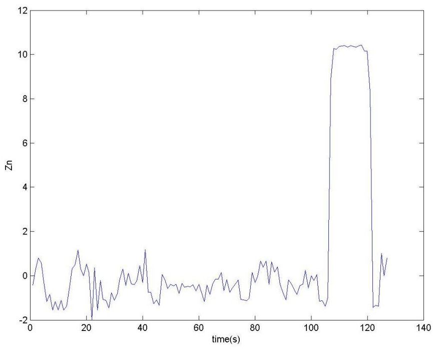

**Fig. 6.** Offset Entropy Statistics $Z_n$

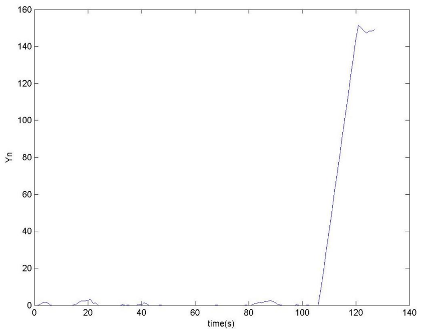

**Fig. 7.** Cumulative Entropy $Y_n$

## 5.3 Time-Based Entropy Detection

In this test, we choose $t = 1s$, $n = 10$, which means that every 1s we compute the entropy value, and judge whether it is the two times of the normal value. If

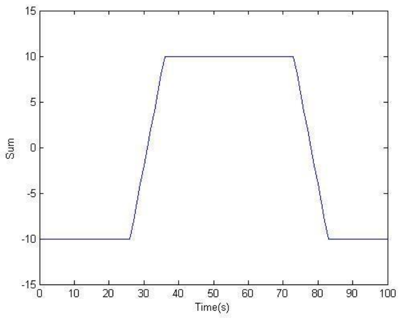

**Fig. 8.** Time-based Entropy Detection

so, we believe that something abnormal happened in this network. If the vector's n elements are all changed to $+1$, which means that the abnormal persisted over $t \times n = 10s$, and we believe that the network is attacked. The test result is shown in Figure 8.

Because here $n = 10$, when the vector becomes $[+1 + 1 \cdots + 1]$, and there are total ten $+1$, we compute the sum equals to 10 in the graph. When the network is completely good, nothing abnormal exists, and the sum then is -10. Of course, the values of $n$ and $t$ depend on the practical environment.

### 5.4 Discussions

From the graphs shown previously, we could see that, for the statistics of entropy, when the attack occurs, there will be a rapid increase for the statistics, and it then reaches a very big value. If the network administrator sets such a value, when exceeding this value, the system will regard it as attack coming. For the cumulative entropy detection approach, we make use of a process to cumulating entropy. We emphasize that the anomaly lasts for a time period, not just happens at a time point. When the attack comes, the system does not immediately give an alarm like the traditional entropy detection method, but the system needs to cumulate a time-period's attack condition, and then gives the judgment. When the network just has some abnormal flow in a normal network environment, our cumulative entropy may not give an alarm. This approach reduces the false alarm rate.

As we use sliding window method to complete the calculation of entropy, the entropy we compute is not all network packets' entropy in a time unit, but just the window size's entropy. Thus we could use another way to judge the anomaly. Set the window size $W = 1000$. From the previous test results, the normal entropy value should be around 2 bits, and the maximum entropy value should be 9.96, when every packet has different IP address. Because we set a small window value, when the attack comes, large number of random IP packets will lead to the rapid increase of entropy, close to 10. During the attack, this value is quite stable. See from the graph, it is approximately a line. In normal traffic case, the entropy always fluctuates, without a stable value. According to this point, we might use a small window size $W$, calculate its maximum entropy $E_{max}$, and when $(E - E_{max}) \to 0$, we say that network is abnormal.

## 6 Conclusions

In this paper, we studied the DDoS detection algorithms based on entropy monitoring, and proposed two improved entropy detection approaches: cumulative entropy and time-based methods. We also conducted experiments for the traditional entropy detection method and the cumulative entropy detection method, respectively. From the test results, we could see that our cumulative entropy detection method has good detection capability. For different network environments, how to configure the threshold value is a key point, which influences the detection efficiency. In the time-based entropy detection method, we introduced a new concept of time cumulating. By setting a system's tolerable detection time, DDoS detection can be carried out without giving a typical threshold value.

## References

1. Shannon, C.E., Weaver, W.: The Mathematical Theory of Communication. University of Illinois Press (1963)
2. Siaterlis, C., Maglaris, V.: Detecting Incoming and Outgoing DDoS Attacks at the Edge using a Single set of Network Characteristics. In: 10th IEEE Symposium on Computers and Communications (ISCC 2005), pp. 469–475 (2005)
3. Chang, R.K.C.: Defending against Flooding-based Distributed Denial-of-Service Attacks: a Tutorial. IEEE Communications Magazine 40(10), 42–51 (2002)
4. Cao, Y., Li, H., Lv, D.: DDoS-based TCP SYN Flood and Defense. Electrical Technology (2004)
5. Dittrich, D.: The Stacheldraht' Distributed Denial of Service Attack Tool (1999), `http://staff.washington.edu/dittrich/misc/stacheldraht.analysis`
6. Risso, F., Delgioanni, L., Varenni, G., Viano, P., Pai, N.: WinPcap: The Windows Packet Capture Library, `http://www.winpcap.org/`
7. Yuan, J., Mills, K.: Monitoring the Macroscopic Effect of DDoS Flooding Attacks. IEEE Transactions on Dependable and Secure Computing 2(4) (2005)
8. Feinstein, L., Schnackenberg, D.: Statistical Approaches to DDoS Attack Detection and Response. In: 2003 DARPA Information Survivability Conference and Exposition (DISCEX 2003), pp. 303–314 (2003)
9. Limwiwatkul, L., Rungsawangr, A.: Distributed Denial of Service Detection using TCP/IP Header and Traffic Measurement Analysis. In: 2004 International Symposium on Communications and Information Technologies (ISCIT 2004), Sapporo, Japan (2004)
10. Lu, J., Yin, C., Zhuang, X., Lu, K., Li, O.: DDoS Attack Detection based on Non-parameter CUSUM. In: Computer and Network (2004)
11. Lin, B., Li, O., Liu, Q.: DDoS Attacks Detection Based On Sequential Change Detection. Computer Engineering 31(9) (2005)
12. Li, Q., Chang, E.-C., Chan, M.C.: On the Effectiveness of DDoS Attacks on Statistical Filtering. IEEE INFOCOM 2005, 1373–1383 (2005)
13. Li, L., Lee, G.: DDoS Attack Detection and Wavelets. In: 12th International Conference on Computer Communications and Networks (ICCCN 2003), pp. 421–427 (2003)
14. Porras, P.A., Neumann, P.G.: EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. In: 1997 National Information Systems Security Conference (NISSC 1997), pp. 353–365 (1997)
15. Jin, S.Y., Yeung, D.S.: DDOS Detection Based on Feature Space Modeling. In: 3rd International Conference on Machine Learning and Cybernetics, Shanghai, pp. 4210–4215 (2004)
16. Jin, S.Y., Yeung, D.S.: A Covariance Analysis Model for DDoS Attack Detection. In: 2004 IEEE International Conference on Communications (ICC 2004), pp. 1882–1886 (2004)
17. Shim, S.-H., Yoo, K.-M., Han, K.-E., Kang, C.-K., So, W.-H., Song, J.-T., Kim, Y.-C.: Destination Address Monitoring Scheme for Detecting DDoS Attack in Centralized Control Network. In: 2006 Asia-Pacific Conference on Communications, pp. 1–5 (2006)

18. Peng, T., Leckie, C., Ramamohanarao, K.: Protection from Distributed Denial of Service Attacks using History-based IP Filtering. In: 2003 IEEE International Conference on Communications (ICC 2003), pp. 482–486 (2003)
19. Lu, W., Traore, I.: An Unsupervised Approach for Detecting DDoS Attacks based on Traffic-based Metrics. In: 2005 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM 2005), pp. 462–465 (2005)
20. Xu, K., Xu, M., Wu, J.: Research on Distributed Denial-of-service Attacks: a Survey. Mini-micro Systems 25(3) (2004)