Acta Technica 62, No. 3B/2017, 603–618 © 2017 Institute of Thermomechanics CAS, v.v.i.

# Time decay to adapt and speedily streaming algorithms for real-time detection of distributed flooding attacks

Hsin-Chang Lin[2,3], Guanling Lee[2,4]

**Abstract.** Network security has become a serious problem. It is getting more and more difficult to find an efficient way of DDoS attacks detection and prevention under the limit of computer's counting of the flows. This paper proposes an efficient data streaming algorithm for real-time and robustly time decay to detect the flooding attacks activity in large networks. The main idea is using a hash-based synopsis data structure while filtering network data streams. This structure can guarantee small space and offer an efficient track as well as accurate synopses. Also, it presents an algorithm for time decay to count the number of potentially malicious connections or packets from the network streams, which focuses on counting the distinct destination as well as source IP by distinguishing different connection types.

**Key words.** Time decay, DDOS attack, flooding attack, attacks detection, counting bloom filter..

## 1. Introduction

### *1.1. Security in network*

There have been many security events in networks that have caught people's attention. Denial of Service (DoS) flooding attacks, which is able to deny users from accessing a specific network resource, have been widely known since the early 1980s. The Computer Incident Advisory Capability (CIAC) reported the first Distributed DoS (DDoS) attack in 1999. As a result, most of the DoS attacks have become distributed in nature, and thus turning to a vital threat to internet security. Followed by is the outbreak of the Code Red worm in July 2001 and The Slammer in

[2] CSIE, National Dong Hwa University, Hualien, 97401, R. O. C.

[3] E-mail: sinchang@gmail.com

[3] E-mail: guanling@gms.ndhu.edu.tw

January 2003 which infected more than 90 % of computers in the Internet within 10 minutes. Ponemon Institute [9] reported that in 2013, the average cyber attack cost an organization \$11.6 million, 26 percent more than in 2012.

### *1.2. DoS attacks*

DoS is a typical network attack when too many concurrent requests overload the system. Without hacking password files or stealing sensitive data, DoS attacks create network congestion by generating a large volume of traffic and causing servers overload in the area of the targeting system. The common way to deliver DoS attack is to disrupt a legitimate user's connectivity by exhausting bandwidth or service resource as well. Anyway, once the attack comes into force, the server can no longer respond to the requests of authorized users. Moreover, not just web servers but also every system connected to internet providing IP network services, such as FTP servers or Mail servers, are also susceptible to DoS attacks. Well-known DoS attacks are the SYN Flood, Teardrop, Smurf, Ping of Death, Land, Black Holes, and the Misdirection.

### *1.3. DDoS flooding attacks*

DDoS flooding attacks is different from ordinary DoS attacks in terms of the number of participants. DDoS attacks involves in an overwhelming amount of attack packets from lots of different sources. Moreover, DDoS attacks direct at one or more targets, such as end-users, web servers, and even entire networks at the same time. The distributed nature of DDoS attacks and the spoofed IP they use to hide their true identity make the attacks extremely difficult to combat and traceback.

Therefore, it is necessary to develop a comprehensive adaptive DDoS defence mechanism which is able to detect appropriately before, during and after the attack. At the [9] reports massive DDoS attacks disable internet access throughout Liberia by hitting the managed DNS provider Dyn on October 21, 2016. The attacks exceeded 500 Gbps.

### *1.4. Network attacks detection*

Current network technology has yet to develop a proper and efficient way to detect and distinguish all DDoS attacks. The paper [12] mentions some challenges in detecting DDoS attacks: (1) large data size, continual data streams and high dimensionality, (2) temporal nature of the data, (3) skewed class distribution, and (4) distributed intrusion detection.

The former Real-time detection techniques which monitor traffic patterns for specific source/destination addresses are impossible to scale when it comes to flooding attacks. Because the set of potential addresses can increase sharply for just a single router in the ISP's backbone or large computer-center, it is infeasible to maintain per-address state smoothly. The detection tech must be able to automatically detect and intercept attacks at incredible high speed and allow large amount of information travelling in complex networks at the same time.

### 1.5. Paper structure

This paper presents an implementation of a cost-effective method with sliding window for CBF which called SWCBF to detect adaptively and send alerts precisely.

The rest of this paper is organized as follows: In Section 2, we discuss background knowledge and related works. Section 3 describes the main ideas and system structure. The experiment results and analysis of it are discussed in Section 4. And finally, Section 5 concludes this work.

## 2. Background and related works

The Internet Protocol (IP) is the network layer protocol of the Internet meant to provide connectionless packet delivery service, however, the service is not reliable, because the delivery of datagram is not guaranteed. Datagram may be lost, duplicated, delayed, or delivered out of order. IP provides a best-effort delivery, packets are not discarded unless resources are exhausted or underlying networks fail.

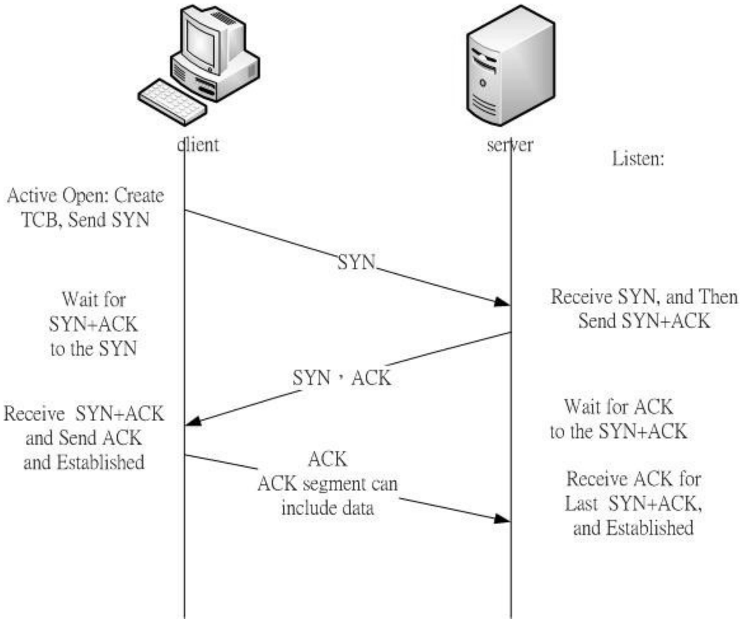

client
server
Listen:
Active Open: Create TCB, Send SYN
SYN
Wait for SYN+ACK to the SYN
Receive SYN, and Then Send SYN+ACK
SYN，ACK
Receive SYN+ACK and Send ACK and Established
Wait for ACK to the SYN+ACK
ACK
ACK segment can include data
Receive ACK for Last SYN+ACK, and Established

Fig. 1. Three-way handshake.

The Transmission Control Protocol (TCP) helps to ensure reliable applications and services. TCP resides between IP and the application layer. It provides a reliable, connection-oriented data stream delivery service.

Fig. 1 show the TCP three-way handshake provides some security against spoof connections. However, it is not perfect. In the three-way handshake process, sequence numbers and acknowledgement numbers are similarly exchanged. The sequence number prediction may allow spoofing, and SYN floods can be used to cause a DoS attack on the machine.

A TCP session is established, which allows the client and the server to synchronize the connection and agree upon the initial sequence numbers. The connection remains open until either the client or the host issues a FIN or RST packet, or the connection times out.

### *2.1. DoS/ DDoS attacks classification*

Network DoS attacks come in many forms. An attacker can either block traffic from clients or flood the server. The public nature of the internet makes it particularly vulnerable to DoS attacks. While DDoS attacks typically involve coopting the services of many other machines to participate in the attack, a Botnet.

Some specific and particularly popular and dangerous types of DoS/ DDoS attacks include [5]:

- **UDP Attacks:** A UDP Attack packet is sent to a random port on the victim system. If the system find out there is no application waiting on the port, it will generate an ICMP packet of destination unreachable to the forged source IP. The common UDP attack takes advantage of the large number of forged UDP packets to cause a great exhaust of the victim system's CPU time, memory, and bandwidth, and eventually break down the system.

- **UDP Attacks:** ICMP Attack (Smurf Attack or Ping Attack)[6]. ICMP is a connectionless protocol for IP operations, diagnostics, and errors. ICMP Flood is a large number of ICMP packets which can overwhelm a target server when it attempts to process each incoming ICMP request, and result in a denial-of-service condition.

- **TCP Flood Attacks:** TCP attack vectors are varied as following: SYN Flood, ACK Flood, SYN+ACK Flood, etc. TCP SYN Flooding is the most commonly used attack with more than 90% of the DoS attacks using it [1]. Some SYN flood mitigation paths open the door for other TCP-based attack vectors. The TCP/IP protocol suite (IPv4) does not readily provide mechanisms to insure the integrity of packet attributes when packets are generated or during end-to-end transmission. A malicious host can exploit the small size of the listen queue by sending multiple SYN requests to a host, but never replying to the SYN+ACK sent back by the other host. By doing so, the other host's listen queue is quickly filled up, and it will stop accepting new connections, until a partially opened connection in the queue is completed or times out.

### *2.2. Detecting and preventing DDoS attacks*

As Fig. 2, once DDoS attacks are detected, the only measure we can take is shutting down the connection between the victim and the internet immediately until the problem is solved. Since the DDoS attacks are extremely aggressive and can consume a large quantity of resource during its' delivery way, researchers are eager to find out a energy-saving way to detect the attacks, stop them as near as possible to the resource, and ultimately, minimize the final damage.

In Fig. 2, it is relatively easier to detect the attack near the victim side rather than the attacker side. However, it is obviously better for the monitor system to discover it near the source of the attacks so as to respond to the attack in time. Consequently, researchers have to make a trade-off between accuracy of the detection and how close to the source the position is.

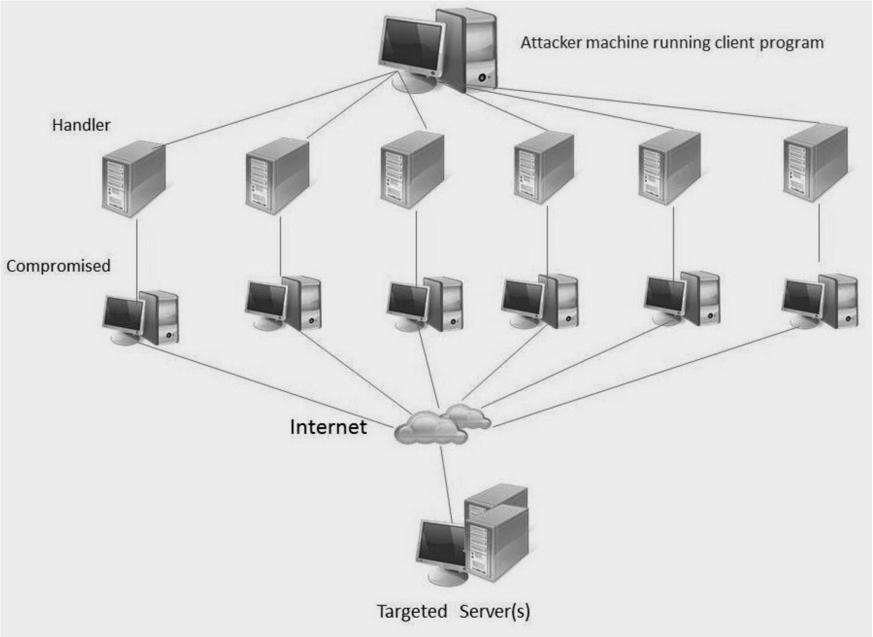


Fig. 2. A DDoS attack.

One way to count DDoS attacks is implementing number prevention techniques. These practices include, but are not limited to strict packet filtering, disabling damaged network services, IP address changing, and regular updating of software.

Generally, DDoS attacks defence mechanisms can be divided into two, i.e. the defence mechanisms against network/transport-level DDoS attacks, and the one against application-level DDoS attacks. The former [25] solution of defence for network/transport-level can be classified into 4 types: source-based, destination-based, network-based, and hybrid; and the defence for application-level is divided into destination-based, and hybrid based on their deployment location.

### 2.3. Network detection architecture

[27] classifies the various detection mechanisms for flooding attacks. Almost all intrusion detection tools follow passive scanning techniques. The main two benefits of passive scanning techniques is the low sensitivity in network environment and a

significantly lighter touch on the network. Compared to the Inline Network Protection (or Active Network Detection), Passive Network Detection provides network administrators with more accurate, prompt, immediate information the instant a system forms and starts working. That is to say, passive scanner is expert in monitoring existing traffic, and thus be able to identify the presence of firewalls, routers, and switches performing NAT, and ultimately characterize the hosts behind them.

### *2.4. To detect DDoS*

In section 2.3, we know that there's definitely some ways to prevent DDoS attacks, including filtering, disabling unused services or setting up network devices in a better way. However, it is a pity that certain attack such as TCP flooding attacks cannot use this way to be completely prevented on the internet, since the TCP flooding attacks consist of internet service protocol which embeds defect mostly. That is to say the IP-spoof made tracing back IP sources impossible.

At present, there are two principal methods for detecting DDoS attacks:

- **Signature-based detection**: Signature detection [16], [20] and [4], [14], [16] search network traffic for a series of bytes or packet sequences defined to be malicious, which means that, the system is unable to identify unknown attacks. Therefore, the system requires signature for every attack, and as the rule set grows, the engine performance slows down inevitably.

- **Anomaly-based detection:** The method observes possible malicious patterns against detecting models. Once a pattern is considered malicious, the security violation is declared immediately. However, how to identify network anomaly in a efficient and accurate way remains a challenging task, considering the ever increasing volumes of network traffic and complexity of usage models. Although sometimes it may prone to false positive, the prominent advantage still makes it irreplaceable. Anomaly detection has a main advantage over signature-based engines in that a new attack can be detected if it falls out of the normal patterns even if the signature doesn't exist. Articles such as [17], [23] provide much more concept about the pros and cons worthy of attention.

Operating with a time-saving and space-saving way, [10] suggested that DoS detection algorithms can guarantee administrator a real time data streaming method. In this paper other two ways of algorithms are given for identifying larger flows-sample and hold and multistage filters, which greatly decrease the memory by taking a constant number of memory references per packet. [26] had proposed a simple way of detecting SYN flooding attack.. Unlike ordinary detection mechanism, they directly detect at leaf routers that connect to hosts end. Based on the protocol behaviour of TCP SYN–FIN pairs, an instance of the Sequential Change Point Detection, the detection is able to immune to flooding attacks on its own.. However, their algorithms must be run on first- or last-mile individual routers, and cannot be used to detect signs of distributed attacks in large networks. Scalability problem also bother a lot in [11], [24] because they require a certain amount of memory to be distribute for each source-destination pair, each source [11], or each flow [24]. [8], [15], [19] presented how to use the Counting Bloom filter to monitor network packet and detect the network attacks.

Most of the networking detection chooses SYN and SYN+ACK pair packets [18], [21] to monitor possible attack in network. However, we found out that RST flag packet is another important attack signal (EX. FIN Scan, Xmas Scan, ACK Scan, etc).

## 3. Bitmap distinct count

### 3.1. Bloom filter and counting bloom filter

Counting the number of distinct networking packet types in network stream has always been a challenging work. Compared to flow-based counters, the bitmap counters have two key properties: 1) low memory usage and 2) provable tradeoffs of memory and accuracy. Bloom filter, a multi-hash function table composed with bitmap-based. A Bloom filter is a simple space-efficient randomized data structure allowing a set of arrays to support membership queries. [2] gives a research on several topics about Bloom filter, including ways to use it, and a unified practical framework for a further understanding of the possible advantages in future applications.

In [2], we briefly a bloom filter represents a set $S$ of $m$ elements from a universe $U$ using an array of $n$ bits, denoted by $B[1], \cdots, B[n]$, initially all set to 0. The filter uses a group $H$ of $k$ independent hash functions $h_1, \cdots, h_k$ with range $\{1, \cdots, n\}$ that independently map each element in the universe to a random number uniformly over the range. For each element $x \in S$, the bits $B[h_i(x)]$ are set to 1 for $1 \le i \le k$. To answer a query of the form "Is the element $y \in S$ ?", we check whether all $h_i(y)$ are set to 1. If not, $y$ is not a member of $S$. If all $h_i(y)$ are set to 1, it is assumed that $y \in S$, and hence a bloom filter may yield a false positive.

The probability of a false positive for an element not in the set is easily derived. If $p$ is the fraction of ones in the filter, it is simply $p^k$. A standard combinatorial argument gives that $p$ is concentrated around its expectation (1).

$$\left(1-(1-1/n)^{mk}\right) \approx \left(1-e^{-km/n}\right) . \tag{1}$$

These expressions are minimized when (2), giving a false positive probability $\boldsymbol{f}$ of $\boldsymbol{f} \approx (1/2)^k \approx (0.6185)^{n/m}$. In practice, $k$ must be an integer, and both $n/m$ (the number of bits per set element) and k should be thought of as constants. For example, when $n/m = 10$ and $k = 7$ the false positive probability is just over 0.008.

$$k = \ln 2 \cdot (n/m) . \tag{2}$$

### 3.2. Counting bloom filter (CBF)

A counting bloom filter (CBF) uses an array of $n$ counters instead of bits. The counters track the number of elements currently hashed to that location [13]. Deletions can now be safely done by decrementing the relevant counters. In this paper used the CBF as Fig. 3, to monitor network packet and detect the network attacks. Unlike the original Bloom filter, the CBF locates each entry on a small counter rather than a single bit. When an item is inserted, the corresponding counters are incremented; when an item is deleted, the corresponding counters are decremented.

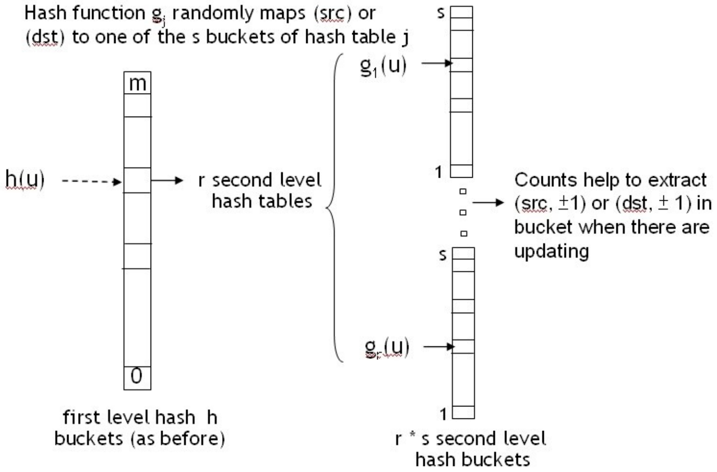
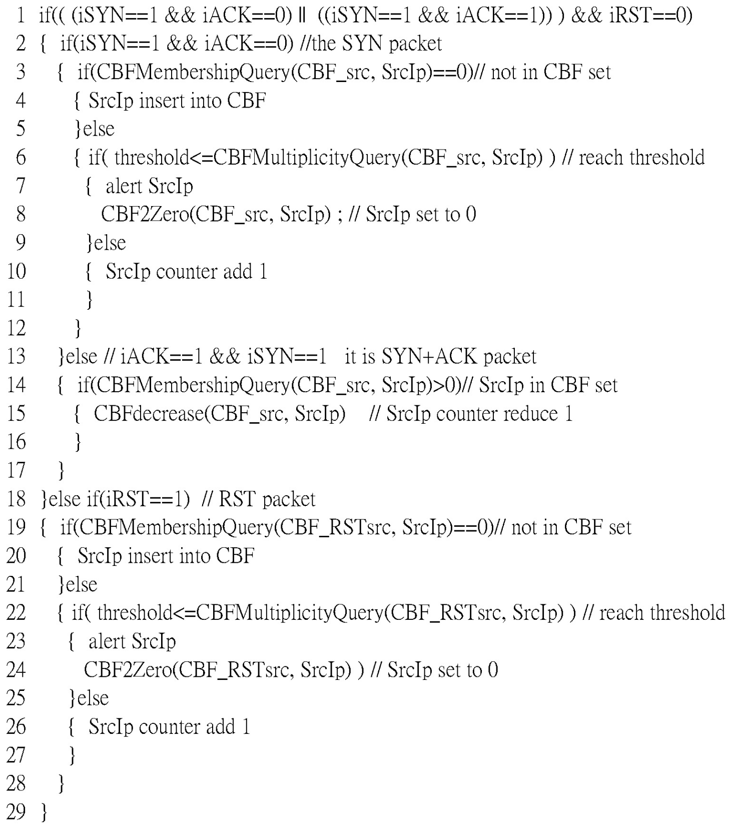

Hash function $g_j$ randomly maps (src) or (dst) to one of the s buckets of hash table j
m
h(u)
r second level hash tables
0
first level hash h buckets (as before)
s
$g_1(u)$
1
Counts help to extract (src, ±1) or (dst, ± 1) in bucket when there are updating
s
$g_r(u)$
1
r * s second level hash buckets

Fig. 3. CBF to count network packet.

In the count structure, each flow update can be abstracted as a triple of the form (src, dest, ±1) where: (1) (src, dest) is a source-destination IP address pair, indicating a flow connection between source and destination. (2) In CBF counter, (src, ±1) and (dest, ±1) indicate the net flow change during the updating process. For example, the SYN packet from source to destination appears with (src, +1) and (dest, +1) in the flow-update stream, and the corresponding ACK packet would appear as (src, –1) and (dest, –1) when successfully connected. (3) When the unwanted query value reaches a threshold set to, a possible malicious flow is recognized as to (src, –n) and (dest, –n).

In CBF packet monitor system, its algorithm as Fig. 4 and this implemented as follows:

1. **CBFadd**: Adds a specified element into the specified CBF, it corresponds to the (src, +1) and (dest, +1) operation when a SYN or RST packet arrives.
2. **CBFdecrease**: Decreases a specified element from the specified CBF, it corresponds to the (src, –1) and (dest, –1) operation when a SYN+ACK pair packet arrives.
3. **CBFelementQuery**: Queries whether a specified element is a member of the set represented by the specified CBF.
4. **CBFmultiplicityQuery**: Queries the multiplicity of a specified element in the multi-set represented by the specified CBF.



```
if(( (iSYN==1 && iACK==0) || ((iSYN==1 && iACK==1)) ) && iRST==0)
{  if(iSYN==1 && iACK==0) //the SYN packet
   {  if(CBFMembershipQuery(CBF_src, SrcIp)==0)// not in CBF set
      { SrcIp insert into CBF
      }else
      { if( threshold<=CBFMultiplicityQuery(CBF_src, SrcIp) ) // reach threshold
        {  alert SrcIp
           CBF2Zero(CBF_src, SrcIp) ; // SrcIp set to 0
        }else
        {  SrcIp counter add 1
        }
      }
   }else // iACK==1 && iSYN==1   it is SYN+ACK packet
   {  if(CBFMembershipQuery(CBF_src, SrcIp)>0)// SrcIp in CBF set
      {  CBFdecrease(CBF_src, SrcIp)    // SrcIp counter reduce 1
      }
   }
}else if(iRST==1)  // RST packet
{  if(CBFMembershipQuery(CBF_RSTsrc, SrcIp)==0)// not in CBF set
   {  SrcIp insert into CBF
   }else
   { if( threshold<=CBFMultiplicityQuery(CBF_RSTsrc, SrcIp) ) // reach threshold
     {  alert SrcIp
        CBF2Zero(CBF_RSTsrc, SrcIp) ) // SrcIp set to 0
     }else
     {  SrcIp counter add 1
     }
   }
}
```

Fig. 4. A CBF packet monitor algorithm.

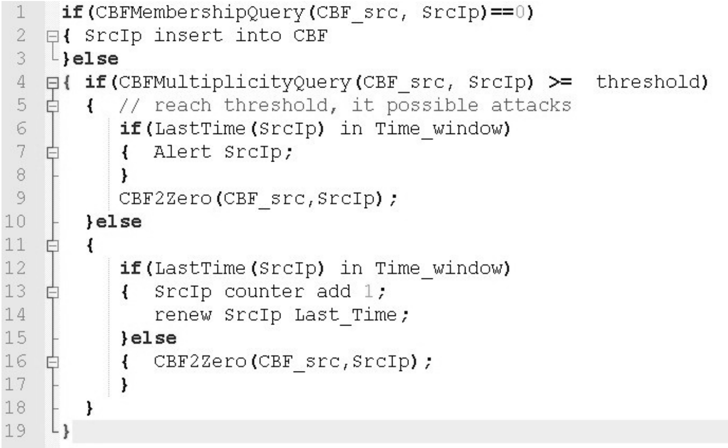
5. **CBF2zero**: Sets a specified reached threshold (n) element from the specified CBF to zero, it corresponds to the (src, –n) or (dest, –n) operation when a SYN or RST packet arrives.

### *3.3. Sliding windows counting bloom filter (SWCBF)*

Most counter papers such as [15] and [25], timeline isn't considered when counting data stream. However, attack monitored 24 hours apart is actually meaningless in network detection, especially for DDoS attacks. Dos attacks happen only when a large amount of malicious flows shows up in a short time. That's why we should keep the time record and clean up the out-of-date data.

Internet data stream are generated continually, making it infeasible to analysis a stream immediately and thoroughly. The best solution is to set a sliding window which is able to maintain the recently arrived data and eliminate the old data outside the sliding window. Only malicious connecting data reserved in the sliding window would be counted, and once those data surpass the threshold, an alarm is sent. After the alarm, the system reset the data and starts a new counting. As the contents of the sliding windows evolve as time goes by, users can receive updated answers.

In CBF packet monitor system, the implemented sliding window CBF which called SWCBF as Fig. 5, it presents our pseudo code for this algorithm.



```
if(CBFMembershipQuery(CBF_src, SrcIp)==0)
{ SrcIp insert into CBF
}else
{ if(CBFMultiplicityQuery(CBF_src, SrcIp) >=  threshold)
  {  // reach threshold, it possible attacks
     if(LastTime(SrcIp) in Time_window)
     {  Alert SrcIp;
     }
     CBF2Zero(CBF_src,SrcIp);
  }else
  {
     if(LastTime(SrcIp) in Time_window)
     {  SrcIp counter add 1;
        renew SrcIp Last_Time;
     }else
     {  CBF2Zero(CBF_src,SrcIp);
     }
  }
}
```

Fig. 5. Pseudo code for SWCBF.

The challenges of the SWCBF system include, amongst others: (1) how to choose the SWCBF bucket size, (2) hash number, (3) how to pick the network packet data, and (4) how to set the sliding window times. [10] told us that 4 bits per counter for each bucket in CBF should suffice for most network applications to avoid counter overflow. Its probability of overflow is no more than $1.37\,\mathrm{m} \cdot 10^{-15}$. But, in real network monitor, the system found the network attack has the same character as similar large packets, which often make 4 bit counter overflow, as showed as in Fig. 9. Therefore, the system adjusted to 8 bits per counter for each bucket in SWCBF.

In this SWCBF system, the $m$ is the number element of the network addressed set that is $256 * 256 * 256 * 256$, $n$ is the number of counter which in our case is $2048/8 = 256$. The number of hash functions, $k$ is chosen according to (2) [13] and the trade-off between the memory cost and false positive. In the case, it had been set to 10.

## 4. Implementation and experiment

### 4.1. SWCBF system for DDoS detection

Fig. 6 presents our system which can monitor the architecture for all network packets at the router as shown in Fig. 6.

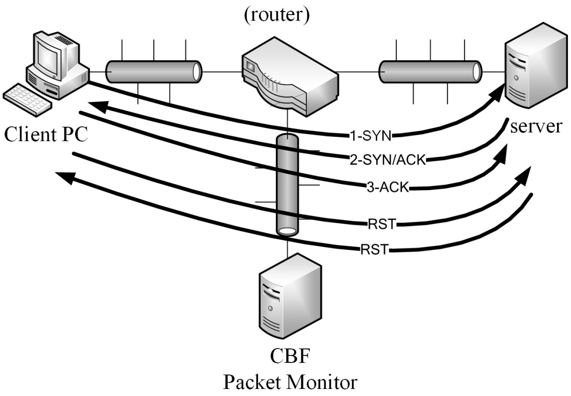
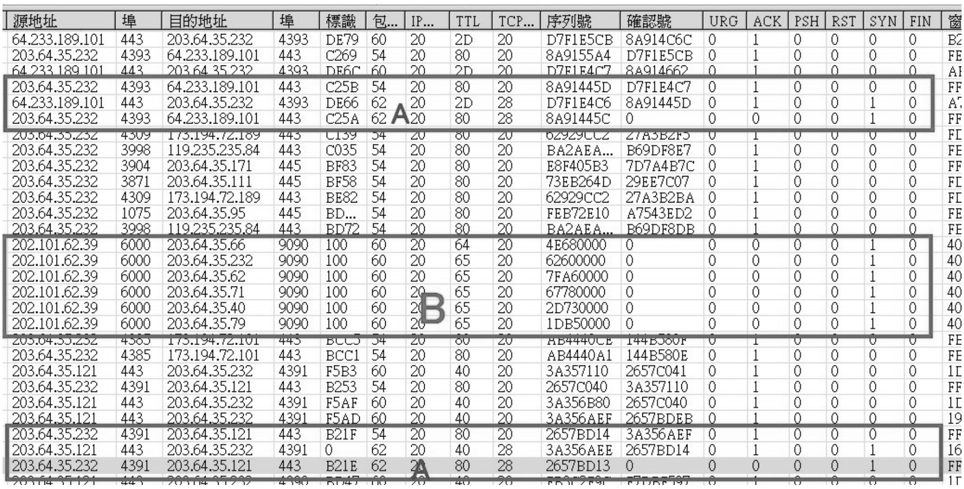


Fig. 6. Architecture for CBF packet monitor.



| 源地址 | 埠 | 目的地址 | 埠 | 標識 | 包... | IP... | TTL | TCP... | 序列號 | 確認號 | URG | ACK | PSH | RST | SYN | FIN | 窗 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 64.233.189.101 | 443 | 203.64.35.232 | 4393 | DE79 | 60 | 20 | 2D | 20 | D7F1E5CB | 8A914C6C | 0 | 1 | 0 | 0 | 0 | 0 | B2 |
| 203.64.35.232 | 4393 | 64.233.189.101 | 443 | C269 | 54 | 20 | 80 | 20 | 8A9155A4 | D7F1E5CB | 0 | 1 | 0 | 0 | 0 | 0 | FE |
| 64.233.189.101 | 443 | 203.64.35.232 | 4393 | DE6C | 60 | 20 | 2D | 20 | D7F1E4C7 | 8A914662 | 0 | 1 | 0 | 0 | 0 | 0 | AF |
| 203.64.35.232 | 4393 | 64.233.189.101 | 443 | C25B | 54 | 20 | 80 | 20 | 8A91445D | D7F1E4C7 | 0 | 1 | 0 | 0 | 0 | 0 | FF |
| 64.233.189.101 | 443 | 203.64.35.232 | 4393 | DE66 | 62 | 20 | 2D | 28 | D7F1E4C6 | 8A91445D | 0 | 1 | 0 | 0 | 1 | 0 | A7 |
| 203.64.35.232 | 4393 | 64.233.189.101 | 443 | C25A | 62 A | 20 | 80 | 28 | 8A91445C | 0 | 0 | 0 | 0 | 0 | 1 | 0 | FF |
| 203.64.35.232 | 4309 | 173.194.72.189 | 443 | C139 | 54 | 20 | 80 | 20 | 62929CC2 | 27A3B2F5 | 0 | 1 | 0 | 0 | 0 | 0 | FD |
| 203.64.35.232 | 3998 | 119.235.235.84 | 443 | C035 | 54 | 20 | 80 | 20 | BA2AEA... | B69DF8E7 | 0 | 1 | 0 | 0 | 0 | 0 | FE |
| 203.64.35.232 | 3904 | 203.64.35.171 | 445 | BF83 | 54 | 20 | 80 | 20 | E8F405B3 | 7D7A4B7C | 0 | 1 | 0 | 0 | 0 | 0 | FF |
| 203.64.35.232 | 3871 | 203.64.35.111 | 445 | BF58 | 54 | 20 | 80 | 20 | 73EB264D | 29EE7C07 | 0 | 1 | 0 | 0 | 0 | 0 | FD |
| 203.64.35.232 | 4309 | 173.194.72.189 | 443 | BE82 | 54 | 20 | 80 | 20 | 62929CC2 | 27A3B2BA | 0 | 1 | 0 | 0 | 0 | 0 | FD |
| 203.64.35.232 | 1075 | 203.64.35.95 | 445 | BD... | 54 | 20 | 80 | 20 | FEB72E10 | A7543ED2 | 0 | 1 | 0 | 0 | 0 | 0 | FE |
| 203.64.35.232 | 3998 | 119.235.235.84 | 443 | BD72 | 54 | 20 | 80 | 20 | BA2AEA... | B69DF8DB | 0 | 1 | 0 | 0 | 0 | 0 | FE |
| 202.101.62.39 | 6000 | 203.64.35.66 | 9090 | 100 | 60 | 20 | 64 | 20 | 4E680000 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 40 |
| 202.101.62.39 | 6000 | 203.64.35.232 | 9090 | 100 | 60 | 20 | 65 | 20 | 62600000 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 40 |
| 202.101.62.39 | 6000 | 203.64.35.62 | 9090 | 100 | 60 | 20 | 65 | 20 | 7FA60000 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 40 |
| 202.101.62.39 | 6000 | 203.64.35.71 | 9090 | 100 | 60 | 20 | 65 | 20 | 67780000 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 40 |
| 202.101.62.39 | 6000 | 203.64.35.40 | 9090 | 100 | 60 | 2B | 65 | 20 | 2D730000 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 40 |
| 202.101.62.39 | 6000 | 203.64.35.79 | 9090 | 100 | 60 | 20 | 65 | 20 | 1DB50000 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 40 |
| 203.64.35.232 | 4385 | 173.194.72.101 | 443 | BCC5 | 54 | 20 | 80 | 20 | AB4440CE | 144B580F | 0 | 1 | 0 | 0 | 0 | 0 | FE |
| 203.64.35.232 | 4385 | 173.194.72.101 | 443 | BCC1 | 54 | 20 | 80 | 20 | AB4440A1 | 144B580E | 0 | 1 | 0 | 0 | 0 | 0 | FE |
| 203.64.35.121 | 443 | 203.64.35.232 | 4391 | F5B3 | 60 | 20 | 40 | 20 | 3A357110 | 2657C041 | 0 | 1 | 0 | 0 | 0 | 0 | 1D |
| 203.64.35.232 | 4391 | 203.64.35.121 | 443 | B253 | 54 | 20 | 80 | 20 | 2657C040 | 3A357110 | 0 | 1 | 0 | 0 | 0 | 0 | FF |
| 203.64.35.121 | 443 | 203.64.35.232 | 4391 | F5AF | 60 | 20 | 40 | 20 | 3A356B80 | 2657C040 | 0 | 1 | 0 | 0 | 0 | 0 | 1D |
| 203.64.35.121 | 443 | 203.64.35.232 | 4391 | F5AD | 60 | 20 | 40 | 20 | 3A356AEF | 2657BDEB | 0 | 1 | 0 | 0 | 0 | 0 | 19 |
| 203.64.35.232 | 4391 | 203.64.35.121 | 443 | B21F | 54 | 20 | 80 | 20 | 2657BD14 | 3A356AEF | 0 | 1 | 0 | 0 | 0 | 0 | FF |
| 203.64.35.121 | 443 | 203.64.35.232 | 4391 | 0 | 62 | 20 | 40 | 28 | 3A356AEE | 2657BD14 | 0 | 1 | 0 | 0 | 1 | 0 | 16 |
| 203.64.35.232 | 4391 | 203.64.35.121 | 443 | B21E | 62 | 2A | 80 | 28 | 2657BD13 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | FF |
| 203.64.35.121 | 443 | 203.64.35.232 | 4390 | [illegible] | 60 | 20 | 40 | 20 | [illegible] | [illegible] | 0 | 1 | 0 | 0 | 0 | 0 | 1D |

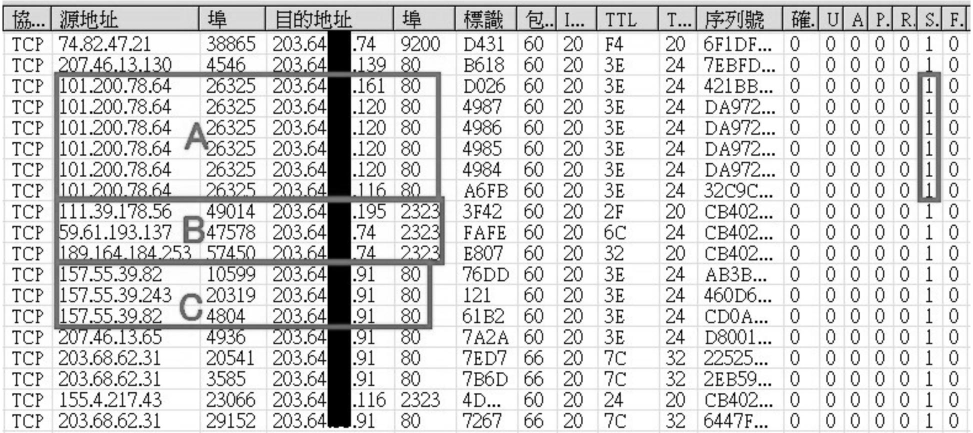
Fig. 7. The raw connection data for SWCBF system.

### 4.2. Running the SWCBF system

Fig. 7 shows the real time, real raw data for our system while monitoring online network connection. The block A is three-way handshake (SYN, SYN+ACK, ACK 3-steps) and the block B is a suspected to SYN attacks. Fig. 8 presents the SYN flood attack in SWCBF system.

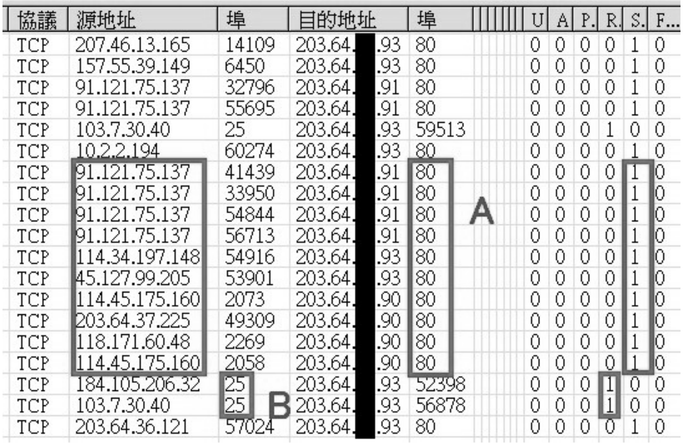
### 4.3. Analysing the data from SWCBF system

In our system, we find a large attacks and other interesting information. In Fig. reffig09, block B shows a continuous generated RST packet at source IP 25 port, which might be a signal of possible internet security problem [22]. Therefore,



| 協... | 源地址 | 埠 | 目的地址 | 埠 | 標識 | 包... | I... | TTL | T... | 序列號 | 確. | U | A | P. | R. | S. | F. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TCP | 74.82.47.21 | 38865 | 203.64[redacted].74 | 9200 | D431 | 60 | 20 | F4 | 20 | 6F1DF... | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 207.46.13.130 | 4546 | 203.64[redacted].139 | 80 | B618 | 60 | 20 | 3E | 24 | 7EBFD... | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 101.200.78.64 | 26325 | 203.64[redacted].161 | 80 | D026 | 60 | 20 | 3E | 24 | 421BB... | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 101.200.78.64 | 26325 | 203.64[redacted].120 | 80 | 4987 | 60 | 20 | 3E | 24 | DA972... | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 101.200.78.64 | 26325 | 203.64[redacted].120 | 80 | 4986 | 60 | 20 | 3E | 24 | DA972... | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 101.200.78.64 | 26325 | 203.64[redacted].120 | 80 | 4985 | 60 | 20 | 3E | 24 | DA972... | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 101.200.78.64 | 26325 | 203.64[redacted].120 | 80 | 4984 | 60 | 20 | 3E | 24 | DA972... | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 101.200.78.64 | 26325 | 203.64[redacted].116 | 80 | A6FB | 60 | 20 | 3E | 24 | 32C9C... | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 111.39.178.56 | 49014 | 203.64[redacted].195 | 2323 | 3F42 | 60 | 20 | 2F | 20 | CB402... | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 59.61.193.137 | 47578 | 203.64[redacted].74 | 2323 | FAFE | 60 | 20 | 6C | 24 | CB402... | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 189.164.184.253 | 57450 | 203.64[redacted].74 | 2323 | E807 | 60 | 20 | 32 | 20 | CB402... | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 157.55.39.82 | 10599 | 203.64[redacted].91 | 80 | 76DD | 60 | 20 | 3E | 24 | AB3B... | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 157.55.39.243 | 20319 | 203.64[redacted].91 | 80 | 121 | 60 | 20 | 3E | 24 | 460D6... | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 157.55.39.82 | 4804 | 203.64[redacted].91 | 80 | 61B2 | 60 | 20 | 3E | 24 | CD0A... | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 207.46.13.65 | 4936 | 203.64[redacted].91 | 80 | 7A2A | 60 | 20 | 3E | 24 | D8001... | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 203.68.62.31 | 20541 | 203.64[redacted].91 | 80 | 7ED7 | 66 | 20 | 7C | 32 | 22525... | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 203.68.62.31 | 3585 | 203.64[redacted].91 | 80 | 7B6D | 66 | 20 | 7C | 32 | 2EB59... | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 155.4.217.43 | 23066 | 203.64[redacted].116 | 2323 | 4D... | 60 | 20 | 24 | 20 | CB402... | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 203.68.62.31 | 29152 | 203.64[redacted].91 | 80 | 7267 | 66 | 20 | 7C | 32 | 6447F... | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

Fig. 8. The SYN flood attack in SWCBF system.



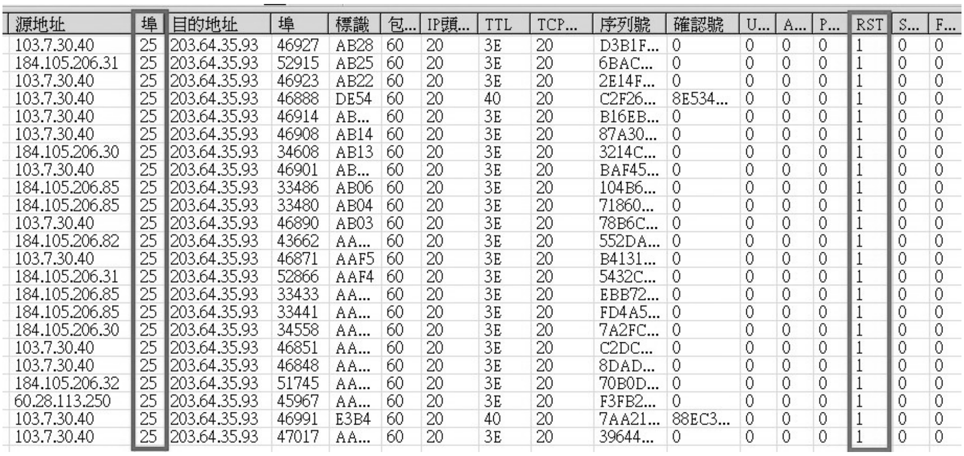
| 協議 | 源地址 | 埠 | 目的地址 | 埠 | U | A | P. | R. | S. | F... |
|---|---|---|---|---|---|---|---|---|---|---|
| TCP | 207.46.13.165 | 14109 | 203.64.[redacted].93 | 80 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 157.55.39.149 | 6450 | 203.64.[redacted].93 | 80 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 91.121.75.137 | 32796 | 203.64.[redacted].91 | 80 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 91.121.75.137 | 55695 | 203.64.[redacted].91 | 80 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 103.7.30.40 | 25 | 203.64.[redacted].93 | 59513 | 0 | 0 | 0 | 1 | 0 | 0 |
| TCP | 10.2.2.194 | 60274 | 203.64.[redacted].93 | 80 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 91.121.75.137 | 41439 | 203.64.[redacted].91 | 80 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 91.121.75.137 | 33950 | 203.64.[redacted].91 | 80 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 91.121.75.137 | 54844 | 203.64.[redacted].91 | 80 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 91.121.75.137 | 56713 | 203.64.[redacted].91 | 80 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 114.34.197.148 | 54916 | 203.64.[redacted].93 | 80 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 45.127.99.205 | 53901 | 203.64.[redacted].93 | 80 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 114.45.175.160 | 2073 | 203.64.[redacted].90 | 80 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 203.64.37.225 | 49309 | 203.64.[redacted].90 | 80 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 118.171.60.48 | 2269 | 203.64.[redacted].90 | 80 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 114.45.175.160 | 2058 | 203.64.[redacted].90 | 80 | 0 | 0 | 0 | 0 | 1 | 0 |
| TCP | 184.105.206.32 | 25 | 203.64.[redacted].93 | 52398 | 0 | 0 | 0 | 1 | 0 | 0 |
| TCP | 103.7.30.40 | 25 | 203.64.[redacted].93 | 56878 | 0 | 0 | 0 | 1 | 0 | 0 |
| TCP | 203.64.36.121 | 57024 | 203.64.[redacted].93 | 80 | 0 | 0 | 0 | 0 | 1 | 0 |

Fig. 9. The SYN flood attack in SWCBF system.

we take a closer look on source IP 25 port, including SYN flood and RST detection. Some distinct possible malware packets are found as Fig. 10. When a special IP is found with a RST flooding attack, we consider it as a TCP reset attacks in general. And obviously, the flooding attack in Fig. 10 has a specific source port (25), which allows us to dig on the attack mechanism further and understand various kinds of internet security issues.

## 5. Conclusions and future work

In this paper, we have proposed a novel algorithm for bloom filter counter which can calculate the number of flows present in the network traffic over sliding windows



| 源地址 | 埠 | 目的地址 | 埠 | 標識 | 包... | IP頭... | TTL | TCP... | 序列號 | 確認號 | U... | A... | P... | RST | S... | F... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 103.7.30.40 | 25 | 203.64.35.93 | 46927 | AB28 | 60 | 20 | 3E | 20 | D3B1F... | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 184.105.206.31 | 25 | 203.64.35.93 | 52915 | AB25 | 60 | 20 | 3E | 20 | 6BAC... | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 103.7.30.40 | 25 | 203.64.35.93 | 46923 | AB22 | 60 | 20 | 3E | 20 | 2E14F... | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 103.7.30.40 | 25 | 203.64.35.93 | 46888 | DE54 | 60 | 20 | 40 | 20 | C2F26... | 8E534... | 0 | 0 | 0 | 1 | 0 | 0 |
| 103.7.30.40 | 25 | 203.64.35.93 | 46914 | AB... | 60 | 20 | 3E | 20 | B16EB... | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 103.7.30.40 | 25 | 203.64.35.93 | 46908 | AB14 | 60 | 20 | 3E | 20 | 87A30... | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 184.105.206.30 | 25 | 203.64.35.93 | 34608 | AB13 | 60 | 20 | 3E | 20 | 3214C... | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 103.7.30.40 | 25 | 203.64.35.93 | 46901 | AB... | 60 | 20 | 3E | 20 | BAF45... | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 184.105.206.85 | 25 | 203.64.35.93 | 33486 | AB06 | 60 | 20 | 3E | 20 | 104B6... | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 184.105.206.85 | 25 | 203.64.35.93 | 33480 | AB04 | 60 | 20 | 3E | 20 | 71860... | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 103.7.30.40 | 25 | 203.64.35.93 | 46890 | AB03 | 60 | 20 | 3E | 20 | 78B6C... | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 184.105.206.82 | 25 | 203.64.35.93 | 43662 | AA... | 60 | 20 | 3E | 20 | 552DA... | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 103.7.30.40 | 25 | 203.64.35.93 | 46871 | AAF5 | 60 | 20 | 3E | 20 | B4131... | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 184.105.206.31 | 25 | 203.64.35.93 | 52866 | AAF4 | 60 | 20 | 3E | 20 | 5432C... | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 184.105.206.85 | 25 | 203.64.35.93 | 33433 | AA... | 60 | 20 | 3E | 20 | EBB72... | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 184.105.206.85 | 25 | 203.64.35.93 | 33441 | AA... | 60 | 20 | 3E | 20 | FD4A5... | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 184.105.206.30 | 25 | 203.64.35.93 | 34558 | AA... | 60 | 20 | 3E | 20 | 7A2FC... | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 103.7.30.40 | 25 | 203.64.35.93 | 46851 | AA... | 60 | 20 | 3E | 20 | C2DC... | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 103.7.30.40 | 25 | 203.64.35.93 | 46848 | AA... | 60 | 20 | 3E | 20 | 8DAD... | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 184.105.206.32 | 25 | 203.64.35.93 | 51745 | AA... | 60 | 20 | 3E | 20 | 70B0D... | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 60.28.113.250 | 25 | 203.64.35.93 | 45967 | AA... | 60 | 20 | 3E | 20 | F3FB2... | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 103.7.30.40 | 25 | 203.64.35.93 | 46991 | E3B4 | 60 | 20 | 40 | 20 | 7AA21... | 88EC3... | 0 | 0 | 0 | 1 | 0 | 0 |
| 103.7.30.40 | 25 | 203.64.35.93 | 47017 | AA... | 60 | 20 | 3E | 20 | 39644... | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

Fig. 10. The Special SYN flood attacks for port 25 in SWCBF.

in an effective way. Our system can not only monitor continuously but be queried constantly and not sacrificing the accuracy. Compared to other algorithms, we have performed a memory-saving and CPU-saving algorithm, ensuring high performance with lower costs.

The core of our project is to set a threshold of timeout window to clear old IP. With the direct bitmaps technique, we can effectively count and hash the index of latest-detected malware IP, record the attacking time and sent an alert if the attack is inside the sliding window.

Most of the counter algorithms are conducted for detecting DDoS attacks targeting at the appearance of large and abnormal connection. However, as internet security issues becoming more widely, different kinds of malicious attacks prosper over time. In the future, we hope that Bloom filter counter model, the lower cost counter, can come into use for detecting other attack mechanisms in addition to DDOS attack. How to identify malicious code not showing a great quantity of flows such as Trojans, spyware or Zombie computer by bloom filter Time decay detection model is our future goal.

## References

[1] Arbor Networks, Inc: *Worldwide Infrastructure Security Report.* Arbor Networks Worldwide Infrastructure Security Report - Volume XI *11* (2016).

[2] A. Broder, M. Mitzenmacher: *Network applications of bloom filters: A survey.* Internet mathematics *1* (2004), No. 4, 485–509.

[3] B. Claise, G. Sadasivan, V. Valluri, M. Djernaes: *Cisco systems NetFlow services export version 9.* Cisco Systems, Request for Comments: 3954, Category: Informational (2004).

[4] G. Carl, G. Kesidis, R. R. Brooks, S. Rai: *Denial-of-service attack-detection techniques.* IEEE Internet Computing *10* (2006), No. 1, 82–89.

[5] HTTP://WWW.CERT.ORG/: *Advisory CA-1996-01 UDP Port Denial-of-Service Attack.* (1996).

[6] HTTP://WWW.CERT.ORG/: *Advisory CA-1998-01 Smurf IP Denial-of-Service Attacks.* (1998).

[7] C. FOSNOCK: *Computer worms: Past, present, and future.* East Carolina University, Greenville, North Carolina, United States (2005).

[8] C. H. SUN, C. C. HU, Y. TANG, B. LIU: *More accurate and fast SYN flood detection.* IEEE International Conference on Computer Communications and Networks, 3–6 August 2009, San Francisco, CA, USA, IEEE Conference Publications (2009), 1–6.

[9] HTTP://MEDIA.SCMAGAZINE.COM/DOCUMENTS/54/2013_US_CCC_REPORT_FINAL_6-1_13455.PDF: *2013 Cost of Cyber Crime Study: United States.* Ponemon Institute, Research Report (2013).

[10] C. ESTAN, G. VARGHESE, M. FISK: *Bitmap algorithms for counting active flows on high speed links.* ACM SIGCOMM Conference on Internet Measurement, 27–29 October 2003, Miami Beach, FL, USA, Proceeding ACM New York (2003).

[11] P. DOKAS, L. ERTOZ, V. KUMAR, A. LAZAREVIC, J. SRIVASTAVA, P. N. TAN: *Data mining for network intrusion detection.* Booktitle: Proc. NSF Workshop on Next Generation Data Mining (2002).

[12] L. FAN, P. CAO, J. ALMEIDA, A. Z. BRODER: *Summary cache: A scalable wide-area web cache sharing protocol.* IEEE/ACM Transactions on Networking *8* (2000), No. 3, 281–293.

[13] G. CARL, R. R. BROOKS, S. RAI: *Wavelet based denial-of-service detection.* Computers & Security *25* (2006), No. 8, 600–615.

[14] S. GANGULY, M. GAROFALAKIS, R. RASTOGI, K. SABNANI: *Streaming algorithms for robust, real-time detection of DDoS attacks.* IEEE International Conference on Distributed Computing Systems (ICDCS), 24–27 June 2007, Toronto, ON, Canada, IEEE Conference Publications (2007), 4–4.

[15] J. NG, D. JOSHI, S. M. BANIK: *Applying data mining techniques to intrusion detection.* IEEE International Conference on Information Technology - New Generations, 13–15 April 2015, Las Vegas, NV, USA, IEEE Conference Publications (2015), 800–801.

[16] J. YANG, H. MA, B. ZHANG, P. CHEN: *An efficient approach for analyzing multidimensional network traffic.* Asia-Pacific Network Operations and Management Symposium, Challenges for Next Generation Network Operations and Service Management, 22–24 October 2008, Beijing, China, Springer LNCS, *5297* (2008), No. 8, 227–235.

[17] L. MIAO, W. DING, J. GONG: *A real-time method for detecting internet-wide SYN flooding attacks.* IEEE International Workshop on Local and Metropolitan Area Networks, 22–24 April 2015, Beijing, China, IEEE Conference Publications (2015), 1–6.

[18] L. Y. WANG, Y. GU, G. WEI: *Detect SYN flooding attack in edge routers.* International Journal of Security and Its Applications *3*(2009), 31–45.

[19] M. LI: *An approach to reliably identifying signs of DDOS flood attacks based on LRD traffic pattern recognition.* Computers & Security *23* (2004), No. 7, 549–558.

[20] M. M. EBADY, A. AMPHAWAN: *Review of syn-flooding attack detection mechanism.* International Journal of Distributed and Parallel Systems *3* (2012), No. 1, 99–117, Cite as: arXiv:1202.1761.

[21] B. MIN, V. VARADHARAJAN: *Design and evaluation of feature distributed malware attacks against the Internet of Things (IoT).* IEEE International Conference on Engineering of Complex Computer Systems (ICECCS), 9–12 December 2015, Gold Coast, QLD, Australia, IEEE Conference Publications (2015), 80–89.

[22] M. LI: *Change trend of averaged Hurst parameter of traffic under DDoS flood attacks.* Computers & Security *25* (2006), 213–220.

[23] N. WEAVER, S. STANIFORD, V. PAXSON: *Very fast containment of scanning worms.* USENIX Security Symposium, 9–13 August 2004, San Diego, CA, USA, Proceedings USENIX Association Berkeley *13*(2004), 3–3.

[24] T. M. Thang, V. K. Nguyen: *Synflood spoof source DDoS attack defence based on packet ID anomaly detection - PIDAD.* Information Science and Applications (ICISA), Springer (LNEE) *376* (2016), 739–751.

[25] H. Wang, D. Zhang, K. G. Shin: *Detecting SYN flooding attacks.* Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, 23–27 June 2002, New York, USA, IEEE Conference Publications *3* (2002), 1530–1539.

[26] M. L. Yu, J. Lavanya, R. Miao: *Software defined traffic measurement with opensketch.* USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2–5 April 2013, Lombard, Il, USA, Proceedings USENIX Association Berkeley (2013), 29–42.

[27] Zargar, S. Taghavi, J. Joshi, D. Tipper: *A survey of defense mechanisms against distributed denial of service ( DDoS) flooding attacks.* Communications Surveys & Tutorials, IEEE 15.4 (2013), 2046–2069.

Received April 30, 2017