Fast detecting Hot-IPs in high speed networks

- **Huynh Nguyen Chinh**
  University of Technical Education Ho Chi Minh City

(Received on December 05 th 2014, accepted on Septemver 23rd 2015)

ABSTRACT

Hot-IPs, hosts appear with high frequency in networks, cause many threats for systems such as denial of service attacks or Internet worms. One of their main characteristics is quickly sending a large number of packets to victims in a short time in network. This paper presents a solution to find Hot-IPs by using non-adaptive group testing approach. The proposed solution has been implemented in combination with the distributed architecture and parallel processing techniques to quickly detect Hot-IPs in ISP networks. Experimental results can be applied to detect Hot-IPs in ISP networks.

Key words: *Hot-IP, denial-of-service attack, Internet worm, distributed architecture, Non-adaptive Group Testing*

INTRODUCTION

Denial of Service attacks and Internet worms

In denial of service (DoS) or distributed denial of service (DDoS) attacks, attackers send a very large number of packets to victims in a very short time. They aim to make an unavailable service to legitimate clients. Internet worms propagate to detect vulnerable hosts very fast in networks [1-2]. The problem is how to fast detect attackers, victims in denial of services attacks and sources of the worms propagating in high speed networks. Based on these results, administrators can quickly have solutions to prevent them or redirect attacks.

There are many methods to detect these risks on network, which are mostly based on Intrusion detection systems/Intrusion prevention systems (IDS/IPS) devices that are allocated before servers to monitor, alert and drop harmful packets. Techniques are used in these solutions that are based on signatures or thresholds. These solutions have some disadvantages in which new attack occurrence and establishing thresholds can decrease the performance of network devices.

High speed networks like ISP which needs a fast solution to decrease these risks. Based on IP traffics going through network devices, every IP packet with its source and destination IP addresses are monitored to appear with a high frequency (Hot-IP), they may be a server that is being attacked. In the case of denial of service attacks [3] or network scanning, attackers send a lot of traffics to a destination in a short time. Routers receive and process a lot of packets in the network. If there are many packets passing through router which have the same IP destination, it may be a DoS attack. In the case of worms [4-5], if there are many packets through the router which have the same source IP address, this host may be infected by worms, and they are scanning the network. Therefore, identifying victims in DoS attacks or Internet worms can be modeled by detecting Hot-IPs.

Our solution aims provides early warning and tracking Hot-IPs by collecting IP packets and finding out Hot-IPs. In our solution, the router acts as a sensor. When a packet arrives at the router, the IP header is extracted and put into groups. Based on the embedded source and destination IP addresses, the analysis is carried out quickly. This method is much faster than one-by-one testing.

ISP network

An ISP is a business or organization that offers users access to the Internet and services. ISP network infrastructure is distributed in areas and hierarchical model. To detect denial of service attacks or Internet worms, ISPs use some techniques, such as based on signatures or features of abnormal traffic behaviors. However, attacker detection is also very important. If we can detect early the identities of the attacker, malicious packets can be dropped and the victim will gain more time to apply attacking reaction mechanisms. Detecting the identities of the attackers requires high state overhead.

In our solution, we use the Non-adaptive Group Testing (NAGT) approach to detect Hot-IPs in networks quickly. It uses low state overhead without requiring either the model of legitimate requests or anomalous behaviors. Besides, ISP architecture is used for early warning Hot-IPs from area to others when it finds out them.

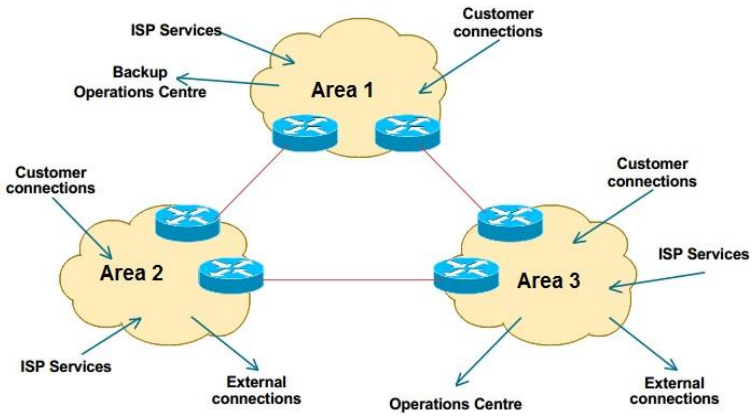


Fig. 1. An ISP network infrastructure

Establishing the distributed architecture to detect worms or denial of service attacks also been studied for many years [8-9]. Detecting risks at an area can help to warn the others early. In the work of Chinh et al. [6-7], they can quickly detect Hot-IPs in network using Non-adaptive Group testing method. This approach can be applied in some applications in data stream, such as: detecting DDoS attackers, Internet worms and networking anomalies.

In this paper, we combine both distributed architecture and NAGT for quickly detecting the Hot-IPs. ISP network architecture is distributed in areas. With this characteristic, we can implement detectors in these areas. Once an area finds out Hot-IPs, it will help other areas to early recognize and supports administrators to have time to find appropriate solutions. In addition, we also implement parallel processing technique to decrease time to detect the Hot-IPs.

We begin with some preliminaries and describe our solution for fast detecting Hot-IPs using NAGT, distributed architecture and parallel processing. The last section is the conclusion.

In this paper, we present a solution for fast detecting Hot-IPs in ISP networks by using Non-adaptive group testing approach with the combination of distributed architecture and parallel processing techniques. We implement strongly explicit *d*-disjunct matrices in our experiment and use network programming to establish the connection between detectors in areas. Once Hot-IPs are detected in one area, it will also immediately alert to other areas.

PRELIMINARIES

Hot-IP

IP address is used to identify host in network. Every packet has an IP header which has source and destination IP addresses. IP packet stream is a sequence of IP packet $a_1, a_2, ..., a_m$ in a link, every packet a_i has an IP address s_i (s_i can be a source address or a destination one depending on particular applications)

Hot-IPs in an IP packet stream are those that appear with a high frequency. Given a IP packet stream of n distinct IP $S = (a_1, a_2, ..., a_m)$, f_i is frequent of IP s_i in S, $f_i = \left|\left\{ j \middle| s_j = s_i \right\}\right|$, $1 \le i \le n$, $1 \le j \le m$, $f_1 + \cdots + f_n = m$. Given a threshold ϕ, Hot-IP = $\left\{ s_i \middle| f_i \ge \phi m \right\}$.

D-disjunct matrix

A binary matrix *M* with *t* rows and *N* columns is called *d*-disjunct matrix if and only if the union of any *d* columns do not contain any other column.

There are three methods to construct d-disjunct matrices [12-14]: greedy algorithm, probabilistic and concatenation codes. To the first two methods, we must save the matrices when the program is running. Therefore, much of RAM space is used in applying these methods because the matrices are often large for the great number of items in high speed networks. Using concatenation codes method, we can generate any columns of the matrix that we need. Therefore, in this paper, we only consider the non-random construction of *d*-disjunct matrix.

Non-random *d*-disjunct matrix is constructed by concatenated codes [14]. The codes concatenating between Reed-Solomon code and identity code is represented below.

Reed-Solomon and codes concatenation

Reed Solomon [15]:

For a message $\bar{\mathbf{m}} = (\bar{\mathbf{m}}_0, ..., \bar{\mathbf{m}}_{k-1}) \in \mathrm{F}_q^k$, let P be a polynomial $P_{\bar{\mathbf{m}}}(X) = \bar{\mathbf{m}}_0 + \bar{\mathbf{m}}_1 X + ... + \bar{\mathbf{m}}_{k-1} X^{k-1}$

In which the degree of $P_{\bar{\mathbf{m}}}(X)$ is at most k-1. RS code $[n,k]_q$ with $k \le n \le q$ is a mapping RS: $\mathrm{F}_q^k \to \mathrm{F}_q^n$ is defined as follows. Let $\{\alpha_1, ..., \alpha_n\}$ be any n distinct members of F_q

$$RS(\bar{\mathbf{m}}) = (P_{\bar{\mathbf{m}}}(\alpha_1), ..., P_{\bar{\mathbf{m}}}(\alpha_n))$$

It is well known that any polynomial of degree at most $k-1$ over F_q has at most $k-1$ roots. For any $\bar{\mathbf{m}} \ne \bar{\mathbf{m}}'$, the Hamming distance between $RS(\bar{\mathbf{m}})$ and $RS(\bar{\mathbf{m}}')$ is at least $d = n - k + 1$. Therefore, RS code is a $[n, k, n-k+1]_q$ code.

Code concatenation [16]:

Let C_{out} be a $(n_1, k_1)_q$ code with $q = 2^{k_2}$ is an outer code, and C_{in} be a $(n_2, k_2)_2$ binary code. Given n_1 arbitrary $(n_2, k_2)_2$ code, denoted by $C_{in}^1, ..., C_{in}^{n_1}$. It means that $\forall i \in [n_1]$, C_{in}^i is a mapping from $\mathrm{F}_2^{k_2}$ to $\mathrm{F}_2^{n_2}$. A concatenation code $C = C_{out} \circ (C_{in}^1, ..., C_{in}^{n_1})$ is a $(n_1 n_2, k_1 k_2)_2$ code defined as follows: given a message $\bar{\mathbf{m}} \in \mathrm{F}^{k_1 k_2} = (\mathrm{F}^{k_2})^{k_1}$ and let $(x_1, ..., x_{n_1}) = C_{out}(\bar{\mathbf{m}})$, with $x_i \in \mathrm{F}_2^{k_2}$ then $C_{out} \circ (C_{in}^1, ..., C_{in}^{n_1})(\bar{\mathbf{m}}) = (C_{in}^1(x_1), ..., C_{in}^{n_1}(x_{n_1}))$, in

which C is constructed by replacing each symbol of C_{out} by a codeword in C_{in}.

In our solution, we choose C_{out} is $[q-1,k]_q$-RS code and C_{in} is identity matrix I_q. The disjunct matrix M is achieved from $C_{out} \circ C_{in}$ by putting all the $N = q^k$ codewords as columns of the matrix. According to [11], given d and N, if we chose $q = O(d \log N),\ k = O(\log N)$, the resulting matrix M is $t \times N$ d-disjunct, where $t = O(d^2 \log^2 N)$. With this construction, all columns of M have Hamming weight equals to $q = O(d \log N)$.

Here is an example of a matrix constructed by concatenated codes.

$$C_{out}: \begin{bmatrix} 0 & 1 & 2 & 0 & 1 & 2 \\ 0 & 1 & 2 & 1 & 2 & 0 \\ 0 & 1 & 2 & 2 & 0 & 1 \end{bmatrix} \qquad C_{in}: \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$C_{out} \circ C_{in}: \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Group Testing

In World War II, millions of citizens in the USA joined the army. At that time, infectious diseases such as syphilis were serious problems. The cost for testing infectors in turn was very expensive and it also took several times. They wanted to detect infected people as fast as possible with the lowest cost. Robert Dorfman [10] proposed a solution to solve this problem. The main idea of this solution was to get N bloods samples from N citizens and combined groups of blood samples to test. It would help to detect infected soldiers using as few tests as possible. This idea formed a new research field: Group testing.

Group testing is an applied mathematical theory applied in many different areas [10]. The goal of the group testing is to identify the set of defective items in a large population of items using as few tests as possible.

There are two types of group testing [11]: Adaptive group testing and non-adaptive group testing. In adaptive group testing, later stages are designed depending on the test outcome of the earlier stages. In non-adaptive group testing, all tests must be specified without knowing the outcomes of the other tests. Many applications, such as data streams, require the NAGT, in which all tests are to be performed at once: the outcome of one test cannot be used to adaptively design another test. Therefore, in this paper, we only consider NAGT.

NAGT can be represented by a $t \times N$ binary matrix M, where the columns of the matrix correspond to items and the rows correspond to tests. In that matrix, $m_{ij} = 1$ means that the j^{th} item belongs to the i^{th} test, and vice versa. We assume that we have at most *d* defective items. It is well-known that if *M* is a d-disjunct matrix, we can show all at most *d* defectives.

NAGT and some analysis

In this subsection, we analysis some features in our solution adapting the requirements in data stream algorithm: one-pass over the input, poly-log space, poly-log update time and poly-log reporting time [12].

We use non-adaptive group testing. Therefore, the algorithm for the hot items can be implemented in one pass. If adaptive group testing is used, the algorithm is no longer one pass. We can represent each counter in $O(\log n + \log m)$ bits. This means we need $O((\log n + \log m)t)$ bits to maintain the counters. With $t = O(d^2 \log^2 N)$ and $d = O(\log N)$, we need the total space to maintain the counters is $O(\log^4 N(\log N + \log m))$. The *d*-disjunct matrix is constructed by concatenated codes and we can generate any column we need. Therefore, we do not need to store the matrix M. Since Reed-Solomon code is strongly explicit, the *d*-disjunct matrix is strongly explicit. D-disjunct matrix *M* is constructed by concatenated codes $C^* = C_{out} \circ C_{in}$, where C_{out} is a $[q,k]_q$-RS code and C_{in} is an identify matrix I_q. Recall that codewords of C^* are columns of the matrix M. The update problem is alike an encoding, in which given an input message $\bar{\mathbf{m}} \in \mathrm{F}_q^k$ specifying which column we want (where $\bar{\mathbf{m}}$ is the representation of $j \in [N]$ when thought of as an element of F_q^k), the output is $C_{out}(\bar{\mathbf{m}})$ and it corresponds to the column $M_{\bar{\mathbf{m}}}$. Because C_{out} is a linear code, it can be done in $O(q^2 \times poly \log q)$ time, which means the update process can be done in $O(q^2 \times poly \log q)$ time. Since we have $t = q^2$, the update process can be finished with $O(t \times poly \log t)$ time. In 2010, P. Indyk et al. [12] proved that they can decode in time $poly(d) \cdot t \log^2 t + O(t^2)$.

RELATED WORK

Finding Hot-IP in IP packets stream is a particular circumstance items in data streams which can represent objects in the network search in high frequency. The items in the data streams can represent sequence queries to an Internet search engine. At that time, high frequent items are commonly searched key words. For Web proxy, these items can be used URL addresses sent from computers in the network. High frequent items are most frequently-asked URL addresses. Routers on the Internet are connected together in order to transfer IP packet streams to the destinations with an immense amount of data. Hot-IPs can be found through these packets. Those Hot-IP may cause problems such as DoS attacks or Internet worms.

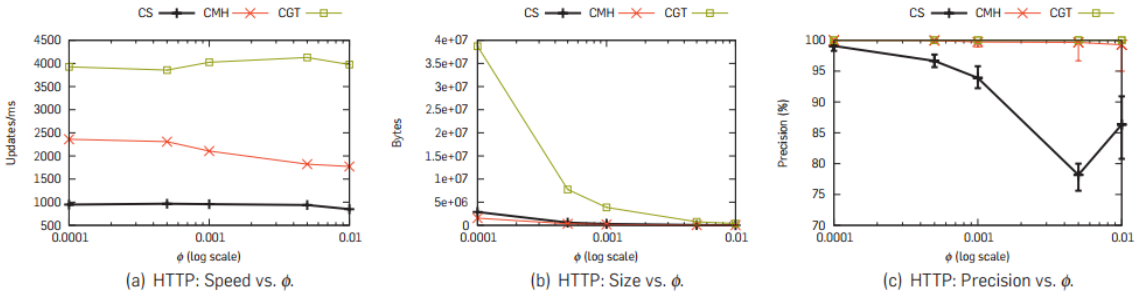
Applications of finding high frequent items in data streams are very important and widespreadly used, therefore many algorithms are suggested. The *Majority* algorithm was proposed by Moore in 1982 [18], the *Frequent* algorithm was proposed by Misra and Gries in 1982 [19], the *LossyCounting* algorithm was proposed by Manku and Motwano in 2002 [20]. The *SpaceSaving* algorithm was introduced in 2005 by Metwally et al [21]. The *CountSketch* algorithm was proposed by Charikar et al. in 2002 [22]. The *CountMin* sketch algorithm was proposed by Cormode and Muthukrishnan in 2005 [23]. Finding frequent items using group testing approach is based on "combinatorial group testing" (CGT) that was proposed by Cormode et al. in 2005.

These algorithms can be divided into two classes: *counted-based* and *sketch-based* algorithms. *Counter-based* algorithms track a subset of items from the input, and the monitor counts the input which is associated with these items. They occupy a great deal of storage space. This is not suitable to quickly detect Hot-IPs established in networks with devices that have limited resources. Therefore, we only consider and compare solutions relating to *sketch-based* algorithms.

Unlike *counter-based* algorithms, *Sketch* ones do not monitor a set of counters of

individual items. On the contrary, these algorithms are linear projections of the input viewed as a vector, and they solve the frequency estimation problem. Therefore they do not explicitly store items from the input. Some algorithms belong to *sketch* such as *CountSketch*, *CountMin*, and *Group Testing*.

These algorithms have been implemented by Cormode et al. in [17], [24]. They use about 10,000,000 HTTP packets and threshold ϕ, $(0.0001 \le \phi \le 0.01)$. Some results are as follows:



CS: CountSketch, **CMH**: CountMin sketch, **CGT**: Cobinatorial Group testing
Fig. 2. Performance of sketch algorithms on real network data [24]

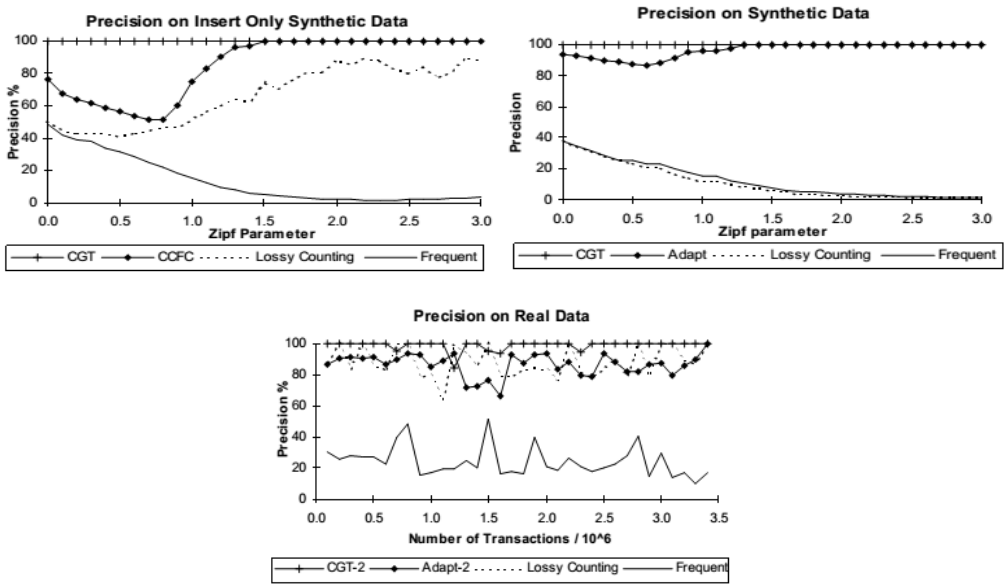


Fig. 3. Performance results on synthetic data and real data [17]

According to the experimental results, group testing method (CGT) consumes a lot of space but it is the fastest *sketch* and is very accurate, with high precision and good frequency estimation in all cases. In this paper, we use some techniques to improve the solution, such as parallel processing and distributed architecture in high speed network.

OUR SOLUTION

A distributed architecture for detecting Hot-IPs

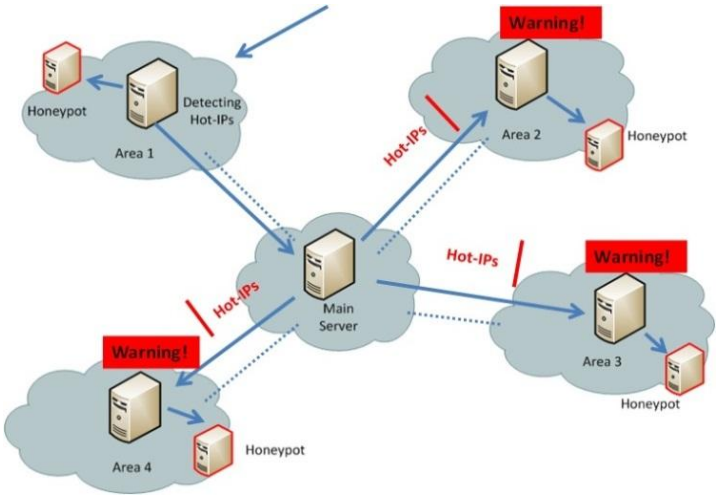


Fig. 4. A distributed architecture for detecting Hot-Ips

It is assumed that ISP network is organized in areas. These areas are connected together. Distributed architecture is used for early warning of some risks on network. For example, if there is a denial of service attack at Area 4 and the victim allocated at Area 2, the detector at Area 4 will send information about the attackers and victims to other areas. From this information, these areas will have some solutions to prevent or limit the attack.

We establish a distributed architecture for fast detecting Hot-IP as follows:

Central server allocated at head quarter and member servers allocated at each area.

Member servers act as sensors periodically to detect Hot-IPs in the network. If they are found, an alert will be sent to central server, all areas, or some areas which contain Hot-IPs. This depends on our purposes.

Central server acts as a sensor and also as a central point to manage all member servers.

The connections between central server and member servers are established out-of-band to transfer information quickly.

Set up

Let N be the number of distinct IP addresses and d be the maximum number of IPs which can be attacked. IP addresses are put into groups (tests) depending on the generation of d-disjunct matrix. The number of tests, $t = O(d^2 \log^2 N)$, is much smaller than N. This means that the total space required is far less than the naïve one-counter-per-IP scheme. With a sequence of m IPs from [N], an item is considered "Hot-IP" if it occurs more than $m/(d+1)$ times [17].

Given the $M_{t\times N} = (m_{ij})$ d-disjunct matrix, $m_{ij} = 1$ if IP_j belonging to the i^{th} group test. Using counters $c_1, c_2, \ldots, c_t$, $(c_i \in [t])$, when an item $j \in [n]$ arrives, incrementing all of the counters c_i such as $m_{ij} = 1$. From these counters, a result vector $r \in \{0,1\}^t$ is defined as follows: $r_i = 1$ if $c_i > m/(d+1)$ and $r_i = 0$, otherwise, a test's outcome is positive if and only if it contains a hot item.

Algorithm 1: Initialization and computing outcome vector

Let:

- *M be d-disjunct* $t \times N$ *matrix*
- $C := (c_1, \ldots, c_t) \in N^t$
- $R := (r_1, \ldots, r_t) \in \{0,1\}^t$
- $IP \in [N]^*$*: sequence of IPs*

We have:

- *For* $i=1$ *to* t *do* $c_i=0$
- *For each* $j \in IP$,
  for $i=1$ *to* t *do*
  if $m_{ij}=1$ *then* $c_i{+}{+}$
- *For* $i=1$ *to* t *do*
  If $c_i > m/(d+1)$ *then* $r_i=1$
  Else $r_i=0$

Detect Hot-IPs

To find Hot-IPs, we use the decoding algorithm.

Algorithm 2: Determining Hot-IPs

Input: ***M*** *be d-disjunct* $t \times N$ *binary matrix and result vector* $\mathbf{R} \in \{0,1\}^t$

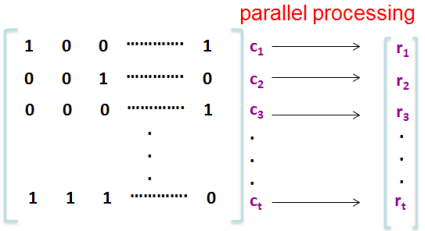
Output: *Hot-IPs*

With each $r_i=0$ *do*
for $i=1$ *to* N *do*
if $(m_{ij})=1$ *Then*
$IP := IP \backslash \{j\}$
Return IP, the set of remaining items

Parallel processing

Parallel processing is a method of having many smaller tasks solving one large problem, so therefore the time required to solve the problem is reduced. In this paper, we run our algorithm solutions in parallel and coordinate their execution.

Parallel processing is used to execute the decoding in our solution as follow. One server acts as a master control, some servers are called slaves. Rows in the matrix *M* are sent to slaves to compute and the results will be sent back to the master. The master collects the outcome values from slaves and then finds Hot-IPs.



In our solution, we use parallel processing model with Parallel Virtual Machine (PVM) to improve the process instead of a single server.

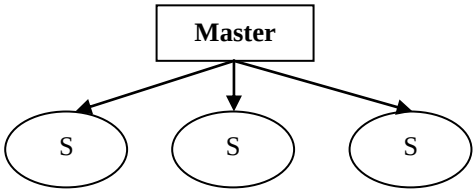


Fig. 5. PVM architecture

PVM is a software environment for heterogeneous distributed computing. It is used to create and access a parallel computing system made from a collection of distributing processors, and treat the resulting system as a single machine. The master is programmed to be responsible for all of the work in the system and the slaves only perform tasks assigned by the master.

The master sends some parameters, such as the matrix M, counters c, and d, to all slaves. These parameters are used for the processing of all slaves. It checks available slaves and sends to them vector M_i (i^{th} test), where M_i refers to i^{th} row. Slaves receive M_j and compute to find out outcome value r_j. Results are sent back to the master. It collects all the values and creates result vector r. From this vector, the master will detect Hot-IPs.

Experimentation

We use four servers to simulate this lab. One at main site is called "Central server" and three servers for three other areas called "Member servers". We use C/C++ network programming in Linux to establish the connection between "Central server" and "Member servers". These servers act as the routers in each area. We use some software from clients to generate any number of packets and implement the algorithm in C/C++, using "pcap" library to capture packets through routers. When each packet is captured, the IP header is extracted. Based on the embedded source and destination addresses, the analysis is done.

We can generate d-disjunct matrices as defined in Section II and support the number of hosts as much as we want. In our experiments, we used 3 matrices which were generated from $[7,3]_8$ - RS code $(d=7, N=4096, t=240)$, $[31,3]_{32}$ - RS code $(d=15, N=32768, t=992)$, and $[31,5]_{32}$ - RS code $(d=7, N=33554432, t=992)$, We tested many cases with different hosts sending packets at the same time, and the results are described in Table 1 (we ignore time to capture packets, we only count the time to decode captured packets).

At each area, member server periodically tracks data streams with the algorithms above. If a Hot-IP is detected, server will send an alert to all other areas, including Hot-IP address.

Table 1. The decoding time for Hot-IPs

RS code	d	Time (s)	N (IPs)
$[15,3]_{16}$	7	0.11	4,096
$[31,3]_{32}$	15	3.65	32,768
$[31,5]_{32}$	7	14.42	100,000

The comparison of decoding time between PVM and single server is described in Table 2. We implement PVM with 3 virtual servers (one master and two slaves).

Master	2 slaves
- Core i5-2410 CPU 2.3 GHz - Memory: 1GB - OS: CentOS	- Intel Pentium 4 CPU 2.4 GHz - Memory: 256 MB - OS: CentOS

Number of IPs: 100,000 – 900,000

Random packets for Hot-IPs: 70-100 million, normal IPs: 300 – 700 packets

Table 2. Decoding time with $[15,5]_{16}$-RS code

N (IPs)	Single server (sec)	PVM (sec)
100,000	154.08	54.16
200,000	154.30	55.24
300,000	166.91	62.02
400,000	167.60	62.75
500,000	189.83	64.48
600,000	219.25	65.32
700,000	236.36	79.33
800,000	261.87	82.97
900,000	308.46	84.41

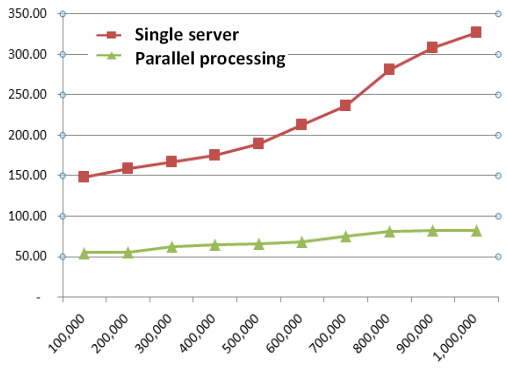


Fig. 6. Single processing and parallel processing

We see that the decoding time to find Hot-IPs is acceptable. We can apply this solution in ISP networks to detect Hot-IPs in reality.

CONCLUSION

Early detection of Hot-IPs in networks is the most important problem in order to mitigate some risks on network. In this paper, we present the efficient solution of the combination of distributed architecture, parallel processing and Non-Adaptive group testing method for speedy Hot-IPs detection in ISP networks. Our future work is to evaluate the solution at ISPs.

Phát hiện nhanh các Hot-IP trong mạng tốc độ cao

- **Huỳnh Nguyên Chính**
  Đại học Sư phạm Kỹ thuật TP. Hồ Chí Minh

TÓM TẮT

Hot-IP là các thiết bị trên mạng hoạt động với tần suất cao, nó là nguyên nhân gây ra các nguy hại cho hệ thống như các tấn công từ chối dịch vụ hay sâu Internet. Một trong những đặc trưng cơ bản của nó là phát tán với số lượng rất lớn các gói tin đến các nạn nhân trên mạng trong một khoảng thời gian rất ngắn. Bài báo này trình bày giải pháp phát hiện nhanh các Hot-IP sử dụng phương pháp thử nhóm bất ứng biến. Giải pháp này được cài đặt kết hợp với kiến trúc phân tán, kỹ thuật xử lý song song để phát hiện nhanh các Hot-IP trong mạng các nhà cung cấp dịch vụ. Kết quả nghiên cứu có thể áp dụng trong mạng của các ISP để phát hiện nhanh các Hot-IP.

Từ khóa: *Hot-IP, tấn công từ chối dịch vụ, sâu Internet, kiến trúc phân tán, thử nhóm bất ứng biến*

REFERENCES

[1]. S. Staniford, D. Moore, V. Paxson, N. Weaver, The top speed of flash worms, *In 2nd ACM Workshop on Rapid Malcode (WORM)*, 33-42 (2004).

[2]. D. Moore, V. Paxon, S. Savaga, C. Shannon, S. Staniford, N. Weaver, The spread of the Sapphire/Slammer worm, *Technical report*, *Caida* (2003).

[3]. T. Peng, C. Leckie, K. Ramamohanarao. Survey of network-based defense mechanisms countering the DoS and DDoS problems, *ACM Computing Surveys*, 39, 1 (2007).

[4]. Z. Chen, L. Gao, K. Kwiat, Modeling the spread of active worms, *In Proceedings of the IEEE INFOCOM 2003*, 1890-1900 (2003).

[5]. G. Serazzi, S. Zanero, Computer virus propagation models, performance tools and applications to networked systems, *Springer Berlin Heidelberg*, 26-50 (2004).

[6]. B.V. Thach, H.C. Nguyen, N.D. Thuc, Early detection for networking anomalies using Non-adaptive Group testing, *ICTC 2013*, 984-987 (2013).

[7]. H.N. Chinh, T. Hanh, N.D. Thuc, Fast detection of DDoS attacks using Non-adaptive Group testing. *IJNSA*, 5, 5, 63-71 (2013).

[8]. Rajab, M. Abu, F. Monrose, A. Terzis. On the effectiveness of distributed worm monitoring. *Proceedings of the 14th USENIX Security Symposium*, . 225-237 (2005).

[9]. Y. Zhang, L.Wang, W. Sun, R.C. Green , A.M. Artificial immune system based intrusion detection in a distributed hierarchical network architecture of smart grid. *Power and Energy Society General Meeting*, 2011 IEEE, 1-8 (2011).

[10]. D. Robert. The detection of defective members of large populations. *The Annals of Mathematical Statistics*, 436-440 (1943).

[11]. D. Dingzhu, F. Hwang. Combinatorial group testing and its applications – 2^{nd}. World Scientific Publishing Company Incorporated (2000).

[12].I. Piotr, N.Q. Hung, A. Rudra. Efficiently decodable nonadaptive group testing. *In Proceedings of the Twenty-First Annual ACMSIAM Symposium on Discrete Algorithms (SODA)*, 1126-1142 (2010).

[13].W. Kautz, S. Roy, Nonrandom binary superimposed codes, *Information Theory, IEEE Transactions on 10*, 4, 363-377 (1964).

[14].N.Q. Hung D.Z. Du, A survey on combinatorial group testing algorithms with applications to DNA library screening, Discrete mathematical problems with medical applications, *Technical Report*, 171-182 (2000).

[15]. I. Reed , G. Solomon, Polynomial codes over certain finite fields, *Journal of the Society for Industrial and Applied Mathematics*, 8, 300–304 (1960).

[16].G.D. Forney Jr, Concatenated codes, *MIT Press* (1966).

[17]. G. Cormode, S. Muthukrishnan, What's hot and what's not: tracking most frequent items dynamically, *In Proceedings of the twentysecond ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, ACM*, 296-306 (2003).

[18].B. Boyer, J. Moore, A fast majority vote algorithm, *Technical Report 35*, Institute for Computer Science, University of Texas (1982).

[19].J. Misra, D. Gries, Finding repeated elements, *Science of Computer Programming*, 143-152 (1982).

[20].G. Manku, R. Motwani, Approximate frequency counts over data streams, *In Proceedings of 28th International Conference on Very Large Data Bases*, 346-357 (2002).

[21].D. Agrawal, A.E. Abbadi, Efficient computation of frequent and top-k elements in data streams, *In International Conference on Database Theory* (2005).

[22].M. Charikar, K. Chen, M.F. Colton, Finding frequent items in data streams, *In Procedings of the International Colloquium on Automata, Languages and Programming (ICALP)*, 693–703 (2002).

[23].G. Cormode, S. Muthukrishnan. An improved Data-stream summary: The Count-min Sketch and its Applications, *Journal of Algorithms*, 55, 58-75 (2005).

[24].G. Cormode, Hadjieleftheriou, M. Finding the frequent items in streams of data, *Communications of the ACM, 52, 10*, 97-105 (2009).