



Analyzing the Number of Varieties in Frequently Found Flows

著者	SHOMURA Yusuke, WATANABE Yoshinori, YOSHIDA Kenichi
journal or publication title	IEICE transactions on communications
volume	E91.B
number	6
page range	1896-1905
year	2008
権利	(c) 2008 IEICE
URL	http://hdl.handle.net/2241/106664

doi: 10.1093/ietcom/e91-b.6.1896

PAPER

Analyzing the Number of Varieties in Frequently Found Flows

Yusuke SHOMURA^{†a)}, Yoshinori WATANABE^{††}, and Kenichi YOSHIDA^{†††},

SUMMARY Abnormal traffic that causes various problems on the Internet, such as P2P flows, DDoS attacks, and Internet worms, is increasing; therefore, the importance of methods that identify and control abnormal traffic is also increasing. Though the application of frequent-itemset-mining techniques is a promising way to analyze Internet traffic, the huge amount of data on the Internet prevents such techniques from being effective. To overcome this problem, we have developed a simple frequent-itemset-mining method that uses only a small amount of memory but is effective even with the large volumes of data associated with broadband Internet traffic. Using our method also involves analyzing the number of distinct elements in the itemsets found, which helps identify abnormal traffic. We used a cache-based implementation of our method to analyze actual data on the Internet and demonstrated that such an implementation can be used to provide on-line analysis of data while using only a small amount of memory.

key words: DDoS, Worm, Scan, P2P

1. Introduction

Abnormal traffic, such as that due to P2P flows, distributed denial of service (DDoS) attacks, and Internet worms, cause various problems in wide-area networks. The vast consumption of network bandwidth caused by P2P mass flows, for example, disturbs smooth communication amongst other users. Network operators therefore need to find and remove these flows in order to maintain optimal Internet operation. Furthermore, DoS/DDoS attacks have become prevalent, and finding Internet worms has also become an important Internet security task.

Various kinds of P2P software, DDoS attacks, and Internet worms are emerging daily; therefore, we need automatic methods of finding them. The method reported in [1] finds P2P flows and Internet worms by looking for frequently found flows because P2P hosts and Internet worms send a lot of packets. A detailed analysis of a group of these flows is also required. When a malicious client tries to find vulnerable servers, it contacts many servers. Malicious clients can thus be found

by analyzing the number of distinct servers that a single client tries to access [2]. This can be done by counting the number of distinct destination IP addresses (DIPs) contained in packets that share the same source IP address (SIP). Implementing such a method is problematic because it requires huge amounts of memory and processing power.

We have therefore developed a simple frequent-itemset-mining method that uses only small amounts of memory and processing power. The most important characteristic of our method is its ability to analyze the number of distinct elements in traffic. It first enumerates frequent itemsets in packet headers and at the same time enumerates the number of varieties in the non-frequent parts of the headers. By analyzing the number of distinct elements in the traffic, various abnormal flows can be found, such as those due to P2P traffic, DDoS attacks, and Internet worms. We used our method to analyze real data on the Internet and the results show that our method can be used for on-line analysis of such data without using large amounts of memory.

The organization of this paper is as follows. Section 2 contains a brief review of related work, and Section 3 explains our method. Section 4 analyzes the characteristics of Internet traffic and shows how we can use our method to find abnormal traffic. Section 5 discusses the resource requirements of our method, Section 6 details some remaining issues, and Section 7 summarizes of our findings.

2. Related Work

Monitoring Internet traffic is an extensively studied area ([3]–[7]). The Internet Engineering Task Force’s (IETF) Interprovider Performance Measurement (IPPM) working group has proposed a framework of IP performance metrics [3]. Their work is important in standardizing the attributes to be measured. Surveyor [4] is a project intended to create a measurement infrastructure. NLANR [5] has a project to develop a large-scale data collection system that will serve as the base infrastructure for various data analyses. MAWI [6] provides an archive of actual Internet traffic data. CAIDA [7] make various tools for analyzing network data. Their visualization tools can be used in various analyses of network data.

Manuscript received January 1, 2003.

Manuscript revised January 1, 2003.

Final manuscript received January 1, 2003.

[†]The author is with the Central Research Laboratory, Hitachi Ltd., Kokubunji-shi, Tokyo, 185-8601, Japan.

^{††}The author is with ALAXALA Networks Corporation, Kawasaki-shi, 212-0058, Japan.

^{†††}The author is with the Graduate School of Business Science, University of Tsukuba, Tokyo, 112-0012, Japan.

a) E-mail: yusuke.shomura.dh@hitachi.com

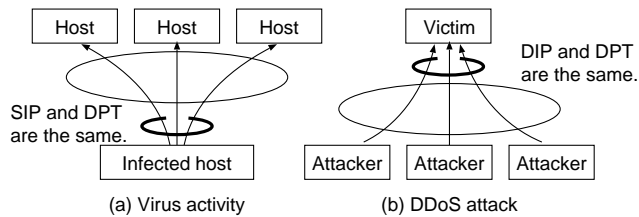


Fig. 1 Behavior of suspicious flows.

The analysis of measured data has also been studied [8], [9]. Some studies (e.g., [1] and [10]) used data-mining techniques to automate analysis. Finding frequently occurring flows is a useful way to analyze Internet traffic because Internet viruses search for vulnerable hosts by repeatedly accessing various IP addresses (Figure 1 (a)). This tends to create many packets that share the same combination of source IP address (SIP) and destination port number (DPT). Here the SIP is the IP address of a host that is already affected by an Internet virus and the affected host is trying to find the next vulnerable host. The DPT is the port number of a target service the virus tries to use in an intrusion. A DDoS attacker attempts to make a computer resource unavailable by flooding it with packets that have a forged sender address (Figure 1 (b)). Such attacks tend to send many packets that share the same combination of destination IP address (DIP) and DPT.

The frequent-itemset-mining algorithm [11] is a promising approach to finding frequently occurring flows caused by infected computers. One of the most important requirements for any method used to find frequently occurring flows is the ability to analyze huge amounts of data without using huge amounts of memory. Among established studies, CPM [1] and Space-Saving [12] can handle vast amounts of data without using huge amounts of memory.

Mori et al. [2] reported that the number of distinct elements in TCP/IP headers was important in network analysis. For example, P2P software generates service port numbers randomly and the packets include various DPTs but the same SIP. Thus, we can find P2P packets by finding groups of packets that share the same SIP but have a variety of DPTs. Several studies have reported methods that collect the number of distinct flows efficiently (e.g., [13], [14], [15], and [16]). For example, the number of distinct SIPs in the flow can be measured by the method proposed in [13]. The number of distinct SIPs for some specific DIP can be measured by the method proposed in [15] and [16]. However, it is not possible to analyze both the distinct elements and mine for frequent itemsets at the same time using either of these methods. Even though such a method enables simultaneous analysis and mining without specifying specific IP or combination of IP and port number, no such method has been developed as of yet.

The method developed by Yoshida et al. [1] can be

used to find the flows of DDoS attacks, scans, and Internet worms. Additional information can be obtained by evaluating the number of distinct elements in TCP/IP headers [2]. We therefore developed a method for collecting statistics on the number of distinct elements in TCP/IP headers without using a large amount of memory.

We also used actual Internet traffic data to evaluate the performance of the mining method we developed. The evaluation showed that cache-based implementation enables data on the Internet to be analyzed in real time without using a huge amount of memory. Note that the huge amount of Internet traffic makes both the analysis of distinct elements in the traffic and the mining of frequent itemsets difficult. Although some researchers have attempted to solve this problem (e.g., [17], [18], and [19]), most of them analyzed only the number of frequently found flows. We have developed a method that can be used to also analyze distinct elements in transactions. Combining these tasks increases the amount of memory required.

Snort [20] is an open source network intrusion detection system that uses a signature-based method. The method we developed is a complement to signature-based methods. Network operators can use our method to screen candidate intrusion events and use a signature-based method to confirm intrusion incidents.

3. Algorithm

To find frequent itemsets and count the number of their corresponding distinct elements, we modified CPM [1], which uses a fixed-size cache memory to find frequent itemsets. Since we intend to control mass flows, our proposed method is first used to find frequent itemsets (i.e. mass flows). Then, the number of varieties are used to classify the found flows. For example, when the modified CPM finds a frequent combination of SIP and DPT, it also counts the other elements in the TCP/IP headers (e.g., DIP and source port number (SPT)). The information about the number of DIPs and SPTs can be used to classify the behavior of the host.

The modified CPM algorithm is shown in Figure 2. The modified parts are underlined. The original CPM algorithm simply counts the frequency of itemsets (i.e., possible combinations of items) in the transactions. In proposed applications, each transaction is made from the TCP/IP header of every single packet. An itemsets function is recursively called to count the frequency of itemsets. Figure 3 shows an example of a function call process with variables. In Figure 3, the precedence numbers appended to the arrow shows the calling sequence. () marks show the value of variable “items”, and {} marks show that of “rests”. A fixed-size cache is used to store the itemsets and their frequencies. Although the original algorithm is extremely simple, it

Algorithm CPM
Variable
Cache[]: fixed-size table
begin
 Create empty cache;
while (input *Transaction*)
 for each *item* in *Transaction*
 Itemsets(*item*, rest of items in *Transaction*);
end

Function Itemsets(*items*, *rests*)
Variable
items[]: items in the current itemsets
rests[]: other items in the transaction
begin
i = index of *items* in cache; (calculated by hash2)
if (*i* is new index)
 increment *cache_diff*[index of original items] by 1;
 increment *cache_cnt*[*i*] by 1;
for each *item* in *rests*
 Itemsets(*items*+*item*, rest of items in *rests*);
if (*cache_cnt*[*i*] ≥ *thresh_hold*)
 report statistics;
 cache_cnt[*i*] = 0;
 cache_diff[*i*] = 0;
end

Fig. 2 Modified CPM algorithm.

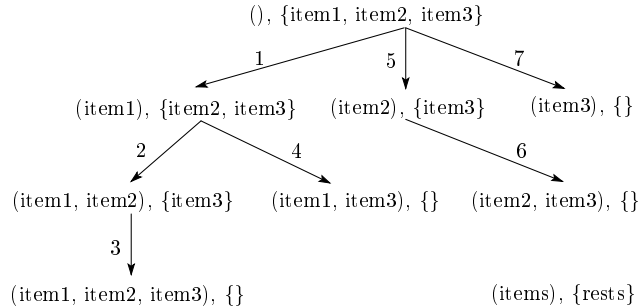


Fig. 3 Function call process.

works well with a specific memory management strategy.

The Hash2 memory management strategy [1] is used to select the cache entry for new items and itemsets (Figure 4). It first calculates “n” hash values of a given item and then generates “n” indexes from these hash values. If the item to be stored is already in the cache, one of the indexes refers to the entry for the item. If the item is a new item, Hash2 selects the index that refers to the least frequent of the “n” entries referred to by the “n” indexes. Then, the old item stored in the cache at that index is replaced by the new item. Here, the “n” influence processing capacity and quantity of mass flows detected. After referring to the recent study by Yoshida et al. [1], we set “n” to 4 in the experiments reported in the next section.

The analysis of Internet traffic data in this work differs from standard frequent itemset mining in that

Function Hash2
Input
Item: Data to be stored in Cache
Variable
Hash[]: Table of Hash Values
Idx[]: Table of Cache Index
begin
 Calculate “n” hash values from *Item*
 and store them into *Hash[]*
Idx[] = *Hash[]* % Cache Size
if (one of entry referred by *Idx[]* stores *Item*)
then **return** *Idx* that refers the entry
else Select *Idx* that refers least frequent entry
 cache_cnt[*Idx*] = 0
 return *Idx*
end

Fig. 4 Pseudo code for memory management.

Table 1 Structure of cache table.

ID	cache _cnt	1st item (e.g. SIP)		2nd item (e.g. SPT)		n-th item
		cache_diff	value	cache_diff	value	
1	16	-	192.168.1.5	-	80	...
2	152	-	10.100.0.19	8	-	...
3	2	2	-	-	443	...
4	40	-	172.16.1.46	-	110	...
5	20	-	192.168.1.5	2	-	...
6	20	2	-	-	80	...

each item in the transaction has a specific meaning, such as SIP and SPT. A cache table with fixed entries is used to store the information (See Table 1). It stores the counters for the distinct elements (*cache_diff* in Table 1) together with the frequency of itemsets (*cache_cnt* in Table 1). The cache table used in our experiments described in the next section has columns for five items: SIP, SPT, DIP, DPT, and protocol number.

Each entry (i.e., each row) in Table 1 stores the value of an item or the number of its distinct elements, that is, each entry stores an alternative value of a specific item (“value” column in Table 1) or the number of its distinct elements (“cache_diff” column in Table 1). When the “value” of an item is stored, the “cache_diff” column of that entry is empty. When the number of distinct elements is stored, the “value” column of that entry is empty. When a cache table for five items is used and the entry stores the information about an itemset with two items, two corresponding columns store the values of items and others store the number of distinct elements of the corresponding items. For example, the first entry in the table (ID=1) shows that

- There are 16 packets whose SIP is “192.168.1.5” and SPT is “80.”

The second entry in the table (ID=2) shows that

- There are 152 packets whose SIP is “10.100.0.19”.

and

- The number of distinct elements in the SPT of

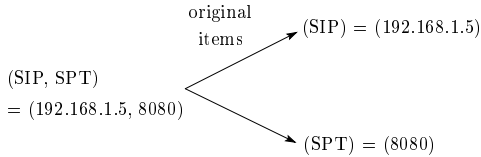


Fig. 5 Original items.

these packets is “8.”

To analyze the distinct elements for the found itemsets, steps (single-underlined in Figure 2) are added. When the method selects an entry to be stored, it checks whether the combination of items is new. When no index in “n” indexes calculated by hash2 refers to the entry, it is determined to be a new item. If it is a new item, the number of distinct elements for the original items is increased by 1. The original items are derived by subtracting any one item from the items processed at that time (Figure 5). For example, if the method makes a new entry with SIP=192.168.1.5 and SPT=80, it increments the counter that stores the number of distinct elements for both item “SIP=192.168.1.5” and “SPT=80” (i.e., SPT cache_diff column of ID=5 and SIP cache_diff column of ID=6 in Table 1).

A normalized operation has also been implemented using the second set of modified steps in our modified CPM algorithm (doubly-underlined in Figure 2). The statistics of frequent items that have a frequency greater than the threshold are reported and then the entry is set to zero. Using this operation, statistics on the same number of related transactions are collected and normalized information about their distinct elements is obtained.

The drawback of using this algorithm is that all the necessary information cannot be held when it is used with a small memory. With a small memory, less frequent items and itemsets must be discarded and not all of them can be found. The number of distinct elements is also overestimated because the novelty check makes errors when data has been discarded. However, we can use this algorithm to analyze extremely biased Internet traffic data. (See Section 5 for details.)

4. Internet Traffic and Abnormal Flows

4.1 Characteristics of Internet Traffic

In the experiments described in this section, we analyzed two traffic logs: one recorded at a monitoring point of a line crossing the Pacific ocean and the other recorded at a monitoring point on a campus line. These data were provided by the MAWI working group [6]. The characteristics of these traffic flows are summarized in Table 2.

Figure 6 shows the number of packets per flow in

Table 2 Traffic characteristics.

	Trans-Pacific line	Campus line
Number of packets	16,031,975	26,954,392
Number of flows	1,228,425	864,916
Percentage of SYN packets	5.27%	2.94%
Average packet length	427.5	677.1

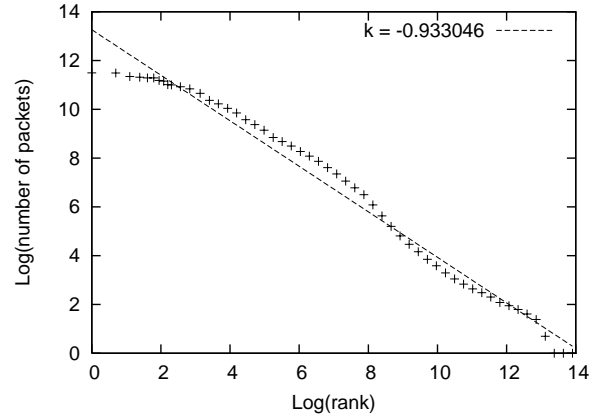


Fig. 6 Number of Packets per Flow.

the Trans-Pacific line. Here, each flow shares the combination of the same source IP address (SIP), destination IP address (DIP), source port number (SPT), destination port number (DPT), and protocol number. As shown in Figure 6, the distribution follows a power law with an exponent of -0.933. The relation between the number of packets P and rank of flows F in the traffic is given by the following equation:

$$P = F^{0.933} \int_1^F x^{-0.933} dx. \quad (1)$$

Although the traffic data involves other N tuples (where $N = 1 \sim 4$), the varieties of flows (i.e., 5 tuples) are dominant; therefore, we can use the number of flows as an index of memory requirements. The duration of each flow is also important. In the following experiments, we analyze every 1000 packets in order to determine the characteristics of each flow. In the Trans-Pacific line traffic, a typical scan takes 451.8 seconds to send 1000 packets and a scan sending packets slowly takes 1363.1 seconds to send 1000 packets. However typical TCP flow takes only 191.8 seconds to send 1000 packets. To analyze the slowest scan, we are required to analyze packets sent during a 1363.1 second period. Since the average packet length is 427.5 bytes, the number P of packets we have to analyze is given by

$$P = 1363.1 * \frac{B}{427.5 * 8} \simeq 0.399 * B, \quad (2)$$

where B is the bandwidth of the communication line (bits per second).

Table 3 Campus line results.

# of different DPTs		# of SIP	collected Flows	flows (%)
1,2	Scan	8	208,040	33.80
	Normal	15	23,192	3.77
3-10	Normal	15	49,818	8.10
11-100	-	-	-	-
100-	P2P	1	1,703	0.28
Total		39	282,753	45.94

Table 4 Trans-Pacific line results.

# of different DPTs		# of SIP	collected Flows	flows (%)
1,2	Scan	10	63,828	11.86
	Normal	5	7,033	1.31
3-10	Normal	6	36,530	6.78
	Unknown	5	7,821	1.45
11-100	Normal	1	7,370	1.37
	Unknown	2	9,933	1.85
100-	P2P	1	3,971	0.74
Total		44	155,580	28.89

Using equation (1) and (2), we can estimate the amount of memory required in order to get the information we need about the traffic packets.

4.2 Analysis of Internet Traffic

A prototype system based on the design described in the previous section has been implemented using a personal computer with a Xeon 2.0 GHz CPU and 2G bytes of RAM. The system reports various piece of information on packet flows appearing more than 1,000 times. Before describing the resource requirements in detail, we give some experimental results obtained using sufficient resources.

4.2.1 Analysis of TCP Packets with SYN flag

We extracted TCP packets that had SYN flags. We analyzed 45.9% of the flows through the campus line and 28.9% of the flows through the Trans-Pacific line. The results are summarized in Tables 3 and 4.

These tables show the number of distinct DPTs for packets that share the same SIP. The number of distinct DPTs for a specific SIP reflects the behavior of the client computer with that IP address, and these tables summarize the types of client computers in the networks. For example, the first row in Table 3 indicates that a few hosts (8 SIPs, third column) generates 208,040 flows to specific DPTs (first column). The 5th row indicates that a single host generates 1,703 flows to various (over 100) DPTs.

The type of flow in the second column can be identified because the number of distinct DPTs indicates

the number of services the SIP host tries to access. The actual service of each flow is confirmed as follows:

Scan

We found that the number of distinct DPTs for these flows is 254 or more than 500.[†] These flows contain only packets with SYN flags and do not contain data packets. These flows indicate typical behavior of crackers or viruses trying to find a target.

P2P

The number of distinct DPTs is more than 100. This behavior indicates that the hosts generate service port numbers randomly and also use a combination of the same TCP port and UDP port. This is typical behavior of a BitTorrent client. We confirmed this by observing the header portion of payloads.

Normal

These are flows that use well-known services, such as SMTP, HTTP, and SSH.

Unknown

Other flows.

The total number of SIPs found in the flows is small. There were 39 SIPs in the campus line and 44 SIPs in the Trans-Pacific line. By analyzing those flows, we found flows with a high probability of being abnormal.

4.2.2 Analysis of all TCP Packets

Other abnormal flows were found by analyzing all the TCP packets. Typical results for the TCP packets analysis are shown in Figure 7. In Figure 7, each point represents the reported number of distinct DPTs (horizontal axis) and DPTs (vertical axis) for specific combinations of SIP and SPT. Since these numbers are reported every time the number of packets reaches 1000 when the proposed method is used, same combinations of SIP and SPT generate multiple points. For example, (d) in the figure indicates that the number of distinct DPTs for packets from a single port of a single host is about 600. The number of distinct DPTs for the same packets is also about 600.

The figure shows the statistics collected for a combination of SIP and SPT, which mainly reflect server output behavior. These statistics can be classified into three groups according to the number of different DPTs.

More than 500 different DPTs:

Four combinations of the same SIP and SPT were found in the Trans-Pacific line data set ((a),(b),(c), and (d) in Figure 7). These four hosts were under

[†]The number of distinct DPTs is also reported simultaneously when using proposed method. Although using the proposed method reports other information, we choose Table 3 and 4 as examples.

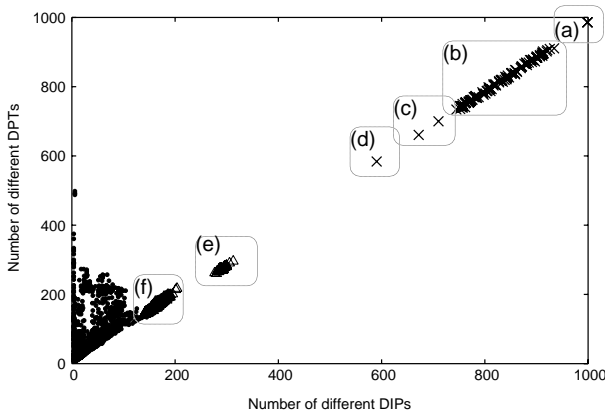


Fig. 7 Characteristics of all TCP packets.

attack or were trying to find vulnerable hosts. Host (a) sent only packets with RST and ACK flags, and it always used SPT 7000. Host (b) sent only packets with SYN and ACK flags, and it also used SPT 7000. Host (c) and (d) sent only packets with SYN and ACK flags, and they used SPT 80.

150-500 different DIPs:

Two combinations of the same SIP and SPT were found in this category (see (e) and (f)). They also used the combination of the same TCP port and UDP port. These two hosts appeared to be hub hosts of an overlay network. They tried to keep sessions active with a large number of hosts in order to maintain an overlay network.

Less than 150 different DIPs:

These are flows about which no conclusion could be made in this analysis.

Clearly, a group of packets with more than 150 different DIPs tends to be suspect and is worth monitoring.

4.2.3 Analysis of UDP Packets

By analyzing UDP packets in a similar way, we found anomalous hosts that scanned vulnerable hosts using UDPs and found hosts that formed a P2P network. Typical results of analysis of the campus line and the Trans-Pacific line are shown in Figure 8 and Figure 9, respectively. Statistics on the packets that shared the same SIP and SPT are shown in these figures.

We also performed a manual analysis of the results from the proposed method. We found seven clusters of varieties of DPTs and DIPs (see Figure 10). The characteristics of each area are summarized in Tables 5 and 6:

Area (1):

Flows in this area were thought to be real-time traffic (voice and video). Manual analysis of flows in this area did not find any flows with unusual behavior.

Area (2):

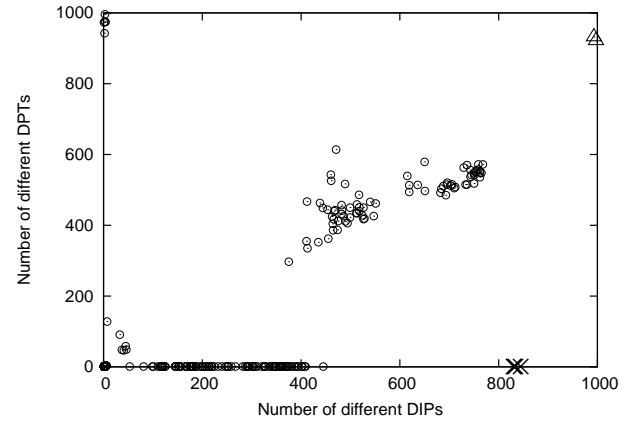


Fig. 8 Campus line (SIP-SPT, UDP).

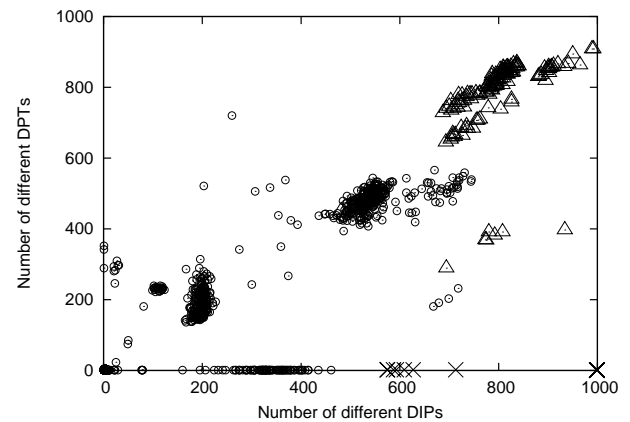


Fig. 9 Trans-Pacific line (SIP-SPT, UDP).

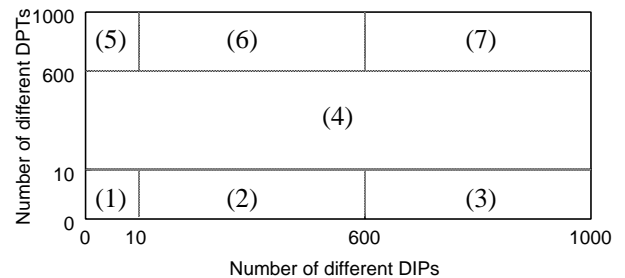


Fig. 10 Seven kinds of DPT and DIP cluster.

Table 5 Campus line results.

Area	# of SIP/SPT Comb.	Collected Packets	Packets (%)	
(1)	Unknown	39	4,545,023	88.74
(2)	DNS	11	115,592	2.26
(3)	Scan, Spam	7	9,751	0.19
(4)	DNS	14	85,005	1.66
(5)	DNS Anomaly	2	5,964	0.12
(6)	DNS	1	3,371	0.07
(7)	P2P	1	2,693	0.05
Total		75	4,767,399	93.09

Table 6 Trans-Pacific line results.

Area		# of SIP/SPT Comb.	Collected Packets	Packets (%)
(1)	Unknown	69	1,313,263	44.75
(2)	DNS	31	112,958	3.85
	Unknown	4	7,073	0.24
(3)	Scan, Spam	22	103,670	3.53
(4)	DNS	26	583,501	19.88
	Broken packets	2	11,003	0.38
	P2P	1	7,319	0.25
	Unknown	1	4,155	0.14
(5)	-	0	0	0.00
(6)	DNS	1	1,600	0.06
(7)	P2P	5	136,518	4.65
Total		162	2,281,060	77.73

Most of the flows in this area were due to DNS client traffic. They consisted of packets going to DPT 53 (i.e., DNS servers). DNS clients connect to multiple DNS servers with the same SPT; therefore, we saw a variety of DIPs in a single combination of the same SIP, SPT, and DPT.

Area (3):

We detected scans and spam in this area. Scans were caused by the SQL slammer worm, which uses port number 1434. Spam used the Windows Messenger service, which uses port numbers 1026 and 1027.

Area (4):

In this area, we detected DNS server traffic and also found broken packets and P2P traffic in the Trans-Pacific data set. Broken packets are located near the Y-axis. P2P traffic is located where $y < x$, which is far from the points of the DNS server.

Area (5):

We detected DNS server traffic that was suspect in this area. By observing the header portion of payloads, we confirmed that these were replies refusing requests for the same name resolution.

Area (6):

We detected DNS server traffic in this area. It consisted of packets from SPT 53.

Area (7):

All the flows in this area were those due to P2P traffic. They used a combination of the same TCP port and UDP port, which is typical behavior of P2P traffic.

Note that the analysis shown in this section follows an un-supervised learning framework. However, the analysis could still find abnormal flows in the Internet traffic. The analysis of the distinct elements in the frequent itemsets contributes to semi-automatic finding of abnormal flows.

5. Evaluation of Resource Consumption

As shown in the previous section, a frequent-itemset-mining method with distinct-element analysis capability can find abnormal flows in Internet traffic. Since combining frequent-itemset-mining and distinct-element analysis increases resource requirements, we detail the resource requirements needed when our proposed method is used.

5.1 Processing capacity

The prototype system took 110.9 seconds to process 1.60 million packets of the Trans-Pacific traffic. If only packets with SYN flags are analyzed, this processing rate (141 k packets per second) is high enough to monitor the traffic in a 9.15-Gbps line (141 kpps * 427.5 byte * 8 bits / 0.0527). It was shown in the previous section that the analysis of packets with SYN flags provides valuable information and that the average use of backbone bandwidth is far less than 100%. Therefore, the performance of our proposed method could potentially handle Internet traffic rates up to 10 Gbps.

5.2 Memory requirement

A cache table with 12 million entries was used in the experiments described in the previous sections. Each entry consumed 50 bytes of memory; therefore, the cache table contains 600 Mbytes of data. With this 600-Mbyte cache table, the results include flows with distinguishing characteristics. Each flow is from a specific SIP and the number of distinct DIPs is 254. Such flows seem to be the flows of worms, which scan C-class address spaces. This result indicates that our proposed method with 12 million entries could count an accurate number of distinct elements in frequently found flows.

The question here is how many packets the proposed method can analyze with a 2-Gbyte memory. If the proposed method can analyze Internet traffic with a 2-Gbyte memory, we can use it as a practical tool for network management. To analyze its performance with 2-Gbyte memory, we evaluated its performance by restricting the number of cache entries.

Figure 11 shows the number of frequently found flows. Although the traffic used in this experiments had about 1800 flows that had more than 1000 packets, a cache table with more than 10k entries can still estimate this number accurately. This signifies that the proposed method can find almost all the frequently occurring flows by using 10k cache entries.

With insufficient memory, the proposed system stores new data by overwriting old cache entries. This causes the second occurrence of old data to be mistaken for the first occurrence of new data, and thus results in the number of distinct elements being overestimated.

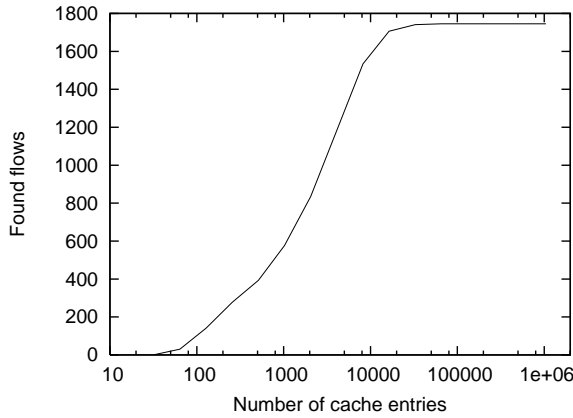


Fig. 11 Found flows.

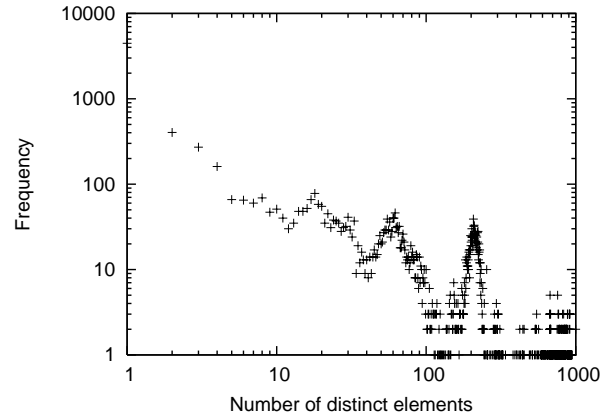


Fig. 13 Distribution of distinct elements (with 1024k entries).

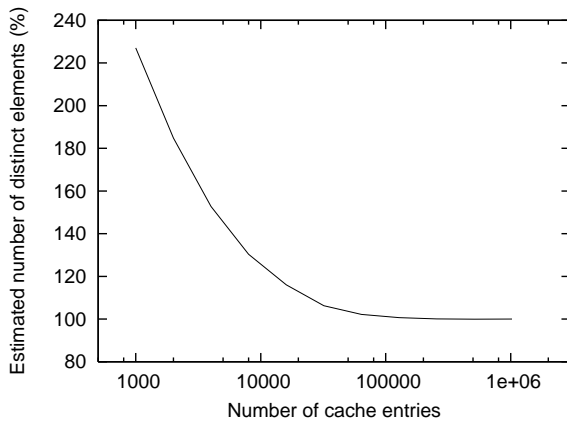


Fig. 12 Estimated number of distinct elements.

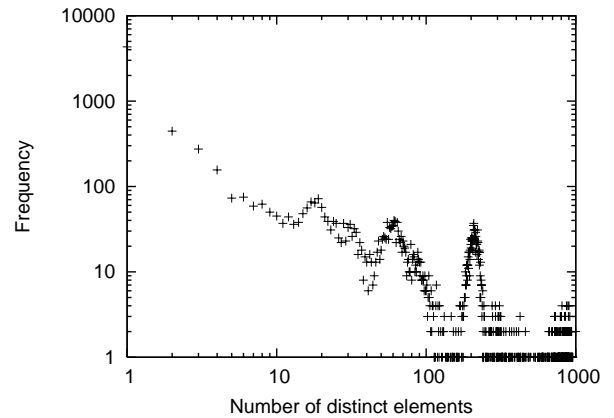


Fig. 14 Distribution of distinct elements (with 32k entries).

Figure 12 shows the average number of DIPs estimated for the TCP packets that share the same SIP.

With 32k cache entries, using our proposed method leads to overestimates in the number of DIPs by 5%, which from a network management viewpoint is acceptable accuracy. Figure 13, 14, and 15 show the captured distribution of the number of DIPs. These figures show that DIP distribution can be captured with 32k~1024k entries but not with 4k entries.

Figure 16 shows results of different experiments. As shown in the previous section, the packets that scan C-class address spaces tend to have longer packet intervals. Since using the proposed method is unsuitable for retaining statistics of flows that have longer packet intervals, we also measured its performance by analyzing specific worms packets. For this purpose, we selected 10 flows that were sent from the same SIPs (i.e. 10 hosts) to C-class addresses and analyzed the results obtained using the proposed method by changing the cache size. If the cache size is sufficient, the proposed method can report accurate statistics. However, if it isn't, it overestimates the number of distinct elements.

Figure 16 shows the number of DIPs that were

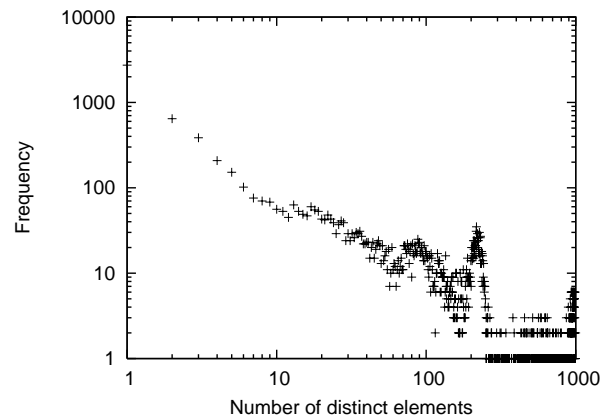


Fig. 15 Distribution of distinct elements (with 4k entries).

found with different cache sizes. With a sufficient cache size (i.e., 1024k entries), the proposed method found the correct number (254) for all 10 flows. With smaller caches, it overestimated the number of DIPs. With 16k cache entries, for examples, it estimated DIP numbers ranging from 514 to 631. We think that even if the number of varieties is overestimated by up to a factor

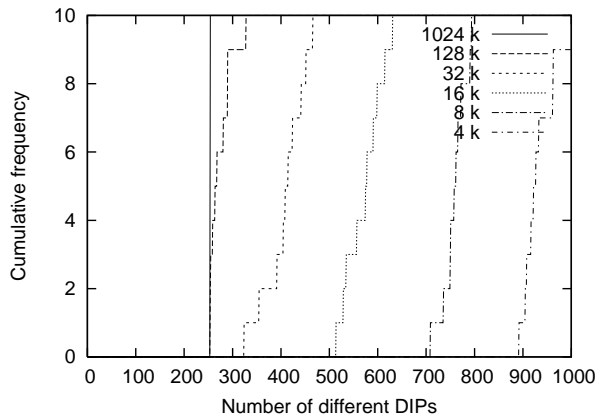


Fig. 16 Distinct elements of worms.

Table 7 Estimated performance.

Assumed bandwidth	10 G bps
Cache entry size	50 bytes
Assumed duration	1363.1 sec.
Number of packets for 1363.1 sec.	3.99G
Number of flows for 1363.1 sec.	267M
Number of entries for 1363.1 sec.	8.2M
Estimated memory size	410 M bytes
Assumed memory size	2 G bytes
Number of entries in 2-Gbyte cache	40 M
Estimated number of flows	1.3G
Estimated number of packets	19.4G
Estimated duration	6737 sec.

of two, using the proposed method will still have applications in the network management. Thus, we have concluded that 30~40k entries (i.e., ~3% of flows) are required for the traffic described in Table 2.

Table 7 shows the estimated traffic that the proposed method can handle. A 10-Gbps Internet line can send 3.99 G packets during 1363.1 seconds (Equation (2)), and that traffic is estimated to contain 267 million flows (Equation (1)). Since the proposed method would require 8.2 million entries to handle that amount of traffic (i.e., about 3% of the flows in it), the required memory size would be 410 Mbytes. If we used a PC with a 2G memory, we could handle traffic for ~ 110 minutes. These results indicate that the proposed method can be used to analyze the Internet traffic in a 10-Gbps line.

5.3 Worst case evaluation

In a worst-case scenario, the estimations of both flow frequency and the number of distinct elements for the flow become large due to the data discard by hash2 memory management. In other words, the memory requirement for the proposed method becomes large if the target network involves more new flows. The CPU requirement is also increased due to the data replacement for the new flows. Note that the CPU requirement for incrementing counters of known flows is much smaller

than that for the data replacement caused by new flows. Thus, both memory and CPU requirements can possibly become larger than those described above.

Although the theoretical analysis of the worst and average case resource requirements is desirable, the difference between the theoretical estimation and real data is too large to design a network monitoring system for daily network monitoring tasks. Some flow characteristics that have not yet been analyzed seem to exist and cause this difference.

Due to this deficiency in real flow characteristics, we evaluated our proposed method by using real Internet traffic in this section.

6. Future Work

We have manually analyzed the flows found by our proposed method; however, the manual analysis was only done to demonstrate the feasibility of the method. In most cases, network operators can use the results of the proposed method to determine potential problems.

In Figures 8 and 9 the data for scans and spam is shown by cross marks and the data for P2P traffic is shown by triangles. Similar to the analysis of TCP packets with the same SIP and SPT, the analysis of the UDP packets that share the same SIP and SPT can identify suspect flows. The data for these kinds of abnormal traffic are clearly found in different parts of these plots.

Analyzing packets with the same SIP and DPT is useful for analyzing client output behavior. Analyzing packets with the same DIP and SPT is useful for analyzing client input behavior. Analyzing packets with the same DIP and DPT is useful for analyzing server input behavior. However, the utility of our proposed method in these kinds of analyses remains to be investigated.

The proposed method might also be useful for daily network monitoring that is not specifically for identifying abnormal flows. For example, Figures 7, 8 and 9 represent some aspect of network usage. They represent the characteristics of the communication between servers and clients. Bandwidth consumption and the percentage of used port numbers are typically observed in daily network monitoring. The additional information provided by our proposed method is expected to facilitate network monitoring.

In this paper, we analyzed every 1000 packets of flow in the experiments. “1000” was chosen empirically. Studying how this number affects the analysis remains as future work.

7. Conclusion

Network management is important for maintaining the Internet as an important social infrastructure. Finding P2P flows, DDoS attacks, and Internet worms is an

important part of network management.

We have developed a method that uses an extremely small amount of memory to analyze the number of varieties in frequently found flows. The most important characteristic of the method is that it can be used to analyze the number of varieties in frequently found flows. In experiments with actual Internet traffic, we demonstrated the following:

- The proposed method can find P2P software, spams, and scans in Internet traffic.
- The performance of the proposed method enables on-line analysis of Internet backbone lines with bandwidths up to 10 Gbps.

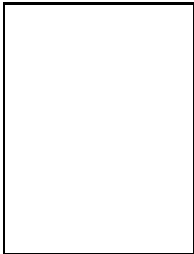
Typical examples of analysis indicate the advantages of our proposed method. Studying the best uses of the method and making full use of its potential remain as future work.

Acknowledgments

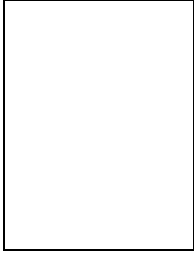
The Internet traffic data analyzed in this study was provided by the WIDE project MAWI working group [6]. We would like to express our thanks for their support.

References

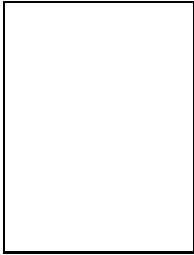
- [1] K. Yoshida, S. Katsuno, S. Ano, K. Yamazaki, and M. Tsuru, "Stream mining for network management," *Transaction of the Institute of Electronics, Information and Communication Engineers*, vol.E89-B, no.6, pp.1774–1780, 2006.
- [2] T. Mori, R. Kawahara, N. Kamiyama, K. Ishibashi, and T. Abe, "Detection of worm-infected hosts by communication pattern analysis," *Technical Report of the Institute of Electronics, Information and Communication Engineers*, pp.1–6, 2005 (in Japanese).
- [3] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis, "Rfc2330, framework for ip performance metrics," 1998.
- [4] "<http://www.advanced.org/surveyor/>."
- [5] "<http://moat.nlanr.net/>."
- [6] Mawi WG, "<http://www.wide.ad.jp/wg/mawi/>," 2007.
- [7] "<http://www.caida.org/>."
- [8] J. Mirkovic, G. Prier, and P.L. Reiher, "Attacking ddos at the source," *Proc. of the 10th IEEE International Conference on Network Protocols*, pp.312–321, 2002.
- [9] Y. Ohsita, S. Ata, M. Murata, and T. Murase, "Detecting distributed denial-of-service attacks by analyzing tcp syn packets statistically," *Proc. of IEEE Globecom 2004*, 2004.
- [10] E.D. Demaine, A. Lopez-Ortiz, and J.I. Munro, "Frequency estimation of internet packet streams with limited space," *Proc. of the 10th Annual European Symposium on Algorithms*, 2002.
- [11] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, ed. J.B. Bocca, M. Jarke, and C. Zaniolo, pp.487–499, Morgan Kaufmann, 12–15 1994.
- [12] A. Metwally, D. Agrawal, and A.E. Abbadi, "Efficient computation of frequent and top-k elements in data streams," *ICDT*, pp.398–412, 2005.
- [13] M. Durand and P. Flajolet, "Loglog counting of large cardinalities," *Proc. European Symposium on Algorithms 2003*, 2003.
- [14] C. Estan, G. Varghese, and M. Fisk, "Bitmap algorithms for counting active flows on high speed links," *Proc. of the 3rd ACM SIGCOMM conference on Internet measurement*, New York, NY, USA, pp.153–166, ACM Press, 2003.
- [15] G. Cormode and S. Muthukrishnan, "Space efficient mining of multigraph streams," *Proc. of the 24th ACM SIGMOD-SIGACT-SIGART symposium on PODS*, pp.271–282, 2005.
- [16] Q. Zhao, A. Kumar, and J. Xu, "Joint data streaming and sampling techniques for detection of super sources and destinations," *Proc. ACM SIGCOMM Internet Measurement Conference*, pp.77–90, 2005.
- [17] N. Hohn and D. Veitch, "Inverting sampled traffic," *Proc. ACM SIGCOMM Internet Measurement Conference*, Miami, USA, Oct. 2003.
- [18] A. Kumar, J. Xu, J. Wang, O. Spatscheck, and L. Li, "Space-code bloom filter for efficient per-flow traffic measurement," *Proc. IEEE infocom*, Hong Kong, March 2004.
- [19] T. Mori, M. Uchida, R. Kawahara, J. Pan, and S. Goto, "Identifying elephant flows through periodically sampled packets," *Proc. ACM SIGCOMM Internet Measurement Conference*, Taormina, Sicily, Italy, Oct. 2004.
- [20] "<http://www.snort.org/>."



Yusuke SHOMURA received B.E. and M.E. degrees in Applied Physics from Waseda University, Japan, in 2001 and 2003. Since joining Hitachi Ltd. in 2003, he has been working on network traffic monitoring.



Yoshinori WATANABE received a B.E. degree in Computer Engineering from Yokohama National University, Japan, in 1987. He joined Hitachi Ltd. in 1987 and has been working at ALAX-ALA Networks Corp. since 2005. His current research interests include network security and traffic monitoring.



Kenichi YOSHIDA received his PhD from Osaka University in 1992. He joined Hitachi Ltd. in 1980 and has been working at the University of Tsukuba since 2002. His current research interests include applications of the Internet and machine-learning techniques.